Analysis of Security and Cryptographic Approaches to Provide Secret and Verifiable Electronic Voting

Stephan Neumann, Jurlind Budurushi, and Melanie Volkamer

TU Darmstadt / CASED, Hochschulstr. 10, 64289 Darmstadt, Germany

Abstract

Electronic voting systems are inextricably bound to security and cryptographic techniques. Over the last decades, countless techniques have been proposed to face the dangers of electronic voting systems with mathematical precision. Certainly, the majority of these works address secrecy and verifiability. In this chapter security and cryptographic techniques are analyzed with respect to those security properties that can be evaluated on the basis of these techniques, namely secrecy, fairness, integrity, and verifiability. Furthermore, we shortly discuss their adequacy to ensure further relevant properties like eligibility and uniqueness, and evaluate security and cryptographic techniques with respect to the costs that come along with their real-world application. We conclude the chapter with a summary of the evaluation results, which can serve as guideline for decision-makers.

Key words: Applied Cryptography, Electronic Voting Requirements, Security Models, Adversary Capabilities, Security Analysis, Electronic Voting Survey.

Introduction

The history of elections reaches back to the ancient Greece and ancient Rome where citizens elected public positions. The implementation of elections has changed over thousands of years from showing of hands to throwing stones and shards into buckets, up to filling paper ballots and throwing them into sealed urns. Since the 1960s, electronic systems are gaining the public interest due to the possible benefits of accurate, fast, and cheap elections. Early electronic voting systems were implemented as voting machines, only since the 1990s, remote electronic voting systems enter the field and turn out to be a promising implementation of absentee voting. Throughout this chapter, we consider only remote electronic voting and use the term electronic voting interchangeably.

Electronic voting systems are inextricably bound to security and cryptographic (SnC) techniques to provide secret, fair, and verifiable elections as well as integrity. Note, SnC techniques considered throughout this work are detached from identification and authentication mechanisms, as this is an orthogonal research direction to this work. Looking back on more than three decades of research, there is a wide range of security and cryptographic techniques striving for secure electronic voting. These techniques are tailored towards special needs and different compromises are made among different properties. Unfortunately, the security model each of the security properties is based on is not specified clearly or is specified in different ways for different approaches. This makes it difficult to compare the different security and cryptographic techniques proposed for secure electronic voting and thus to decide which is appropriate for a special type of election. This gap is addressed within this chapter. Thereby, we support decision-makers in finding adequate SnC techniques to implement electronic voting with respect to their targeted electoral circumstances.

We focus our analysis on security and cryptographic techniques. Correspondingly, the focus is on those security properties which these techniques can already provide without combining them with identification and authentication techniques and without building the whole voting system. These are: secrecy, fairness, integrity, and verifiability. The concrete definitions of these security properties were derived within an interdisciplinary project between legal and technical scientists. These definitions are provided in this chapter. In addition, we developed a common modular security model allowing us to deduce the degree of fulfillment of these properties for concrete SnC techniques. This security model contains an exhaustive list of adversarial capabilities which were deduced from the literature. This security model is presented in this chapter. We, afterwards, select well known SnC

techniques for electronic voting systems from the literature and evaluate them with respect to their security model. Moreover, we shortly discuss the SnC techniques' adequacy to satisfy further security properties namely eligibility and uniqueness when combined with corresponding identification and authentication techniques, as well as the costs to apply these techniques within real-world applications.

Before diving into the main sections of this chapter, we added a background section. Here, we review the related work, provide an overview of the components involved in the electronic voting process, and the preliminaries required in the remainder of this work. We conclude the chapter with a summary of our work and point the reader to future research directions in the electronic voting community.

Background

The first part of this section reviews related literature and shows where the present work is settled in the current state of the art on SnC techniques. In the second part, we outline components generally involved in the electronic voting process. Afterwards, we provide the preliminaries used throughout the analysis. More precisely, the preliminaries cover secret sharing techniques, encryption schemes, digital signature schemes, zero-knowledge proof systems, and the Benaloh challenge. The reader familiar with these preliminaries can safely skip these parts.

Related Work

In this subsection, we review comparative surveys and analyses of SnC techniques in electronic voting systems and settle our own contribution. In (Rjašková, 2002), the author gives a comprehensive overview on cryptographic primitives used in electronic voting and reviews cryptographic voting protocols laying the foundations for her own receipt-free protocol. Due to her own goal, the main focus of her work is receipt-freeness, i.e., secrecy under special adversarial capabilities while our analysis also addresses fairness, vote integrity and verifiability. In (Smith, 2005), the author provides a comprehensive overview on cryptographic primitives and techniques used in electronic voting. Diving into great mathematical detail, the author aims at providing technical background for theoretical cryptographic electronic voting schemes. However, in his work, the author focuses on cryptographic questions such as the computational complexity to compute certain operations. Both, primitives and techniques, are however neither analyzed against legally-derived criteria nor based on a common security model. Lambrinoudakis, Tsoumas, Karyda, and Ikonomopoulos (2003) published an overview work on security techniques underlying electronic voting systems. Both, the classification of these techniques and their analysis does not build upon clear methodologies but rather focuses on providing a basic understanding of these protocols. As opposed to their work, our chapter focuses on a methodological approach in the classification and analysis of SnC techniques, which helps utilizing our results by decision-makers. MacNamara and Iedemska (2012) provide an overview work on cryptographic techniques underlying electronic voting systems. The authors analyze blind signatures, homomorphic encryption, and mix-nets. The declared properties are however not strictly related to the analysis of the techniques such that the analysis and its final conclusions remain vague. Mursi, Assassa, Abdelhafez, and Samra (2013) recently published a survey in which security techniques underlying electronic voting systems are shortly presented and comparatively analyzed. Due to their broad set of security requirements derived from the literature, the analysis of these techniques remains abstract. As opposed to their work, the goal of this work is to provide security models of SnC techniques with respect to these properties that can be evaluated on the basis of these SnC techniques.

Components

Usually, the following entities contribute to the overall electronic voting process: An entity is declared to be a *voter* if her identity is contained in the electoral roll. The *registration authority* is in charge of authorizing eligible voters to cast votes. As such, the registration authority holds the electoral roll. The *tallying authority* is the entity in charge of processing cast votes in order to tally the election result. The *key trustee* is an optional entity holding a secret key. In particular, authorities and

trustees are often distributed such that the overall process can be delegated to a set of entities in order to incorporate stronger security models. The electronic voting system usually relies on one further component, namely the *bulletin board*. It is a server component to which everyone has read access and each authorized entity has corresponding write access. The *voting environment* consists of the hardware as well as the operating system and browser used by the voter to cast her vote.

Secret Sharing

Secret sharing allows splitting a secret apart such that individual shares do not allow conclusions about the secret but a set of shares allows one to reconstruct the secret.

Specification

A secret sharing scheme is a tuple of algorithms (S, R), where S is the sharing algorithm, R the reconstruction algorithm.

A simple secret sharing scheme can be implemented by the XOR ($\oplus$) operator. Assume a dealer wants to share secret s among n participants. Then the dealer randomly draws $s_1, \dots, s_{n-1}$ and computes s_n, such that.

$$s = s_1 \oplus \dots \oplus s_{n-1} \oplus s_n$$

The dealer provides shareholder i with s_i. If all shareholders release their shares, they can reconstruct s according to the above definition. One drawback (amongst others) of this technique is that all shares are needed to reconstruct the shared secret.

Shamir / Feldman Secret Sharing

In contrast to the simplest form of secret sharing, a (t, n) *threshold secret sharing* allows reconstructing the secret having $t < n$ shares. In (Shamir, 1979), the dealer randomly draws values $r_1, \dots, r_{t-1}$ and generates polynom of degree t of the following form

$$f(x) = s + r_1 x + r_2 x^2 + \dots + r_{t-1} x^{t-1}$$

The dealer computes key shares $f(1), \dots, f(n)$ and provides each participant i with her share $(i, f(i))$ for $i \in \{1, \dots, n\}$. According to the fundamental theorem of algebra, for an arbitrary t-set of shares $(i, f(i))$, the polynom $f(x)$ can be reconstructed by the Lagrange interpolation:

$$f(x) = \sum_{i=0}^{t-1} f(i) \cdot \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

The secret s is given by the equation $s = f(0)$.

Shamir's scheme relies on a trusted *dealer* that has to split the secret properly; otherwise corrupt shares cannot be identified and composing distinct sets of t shares would result in distinct reconstructed values. In verifiable secret sharing schemes, the dealer has to provide proofs that the issued secret shares allow to reconstruct the secret afterwards. One technique to extend Shamir's scheme has been proposed by Feldman (Feldman, 1987). Assume two large primes q, p are given such that $q|(p-1)$ and a generator g of order q. The dealer after generating polynom $f(x)$ commits on this polynom by publishing

$$g^s \bmod p, g^{r_1} \bmod p, \dots, g^{r_{t-1}} \bmod p.$$

Whenever the dealer issues a share to a shareholder i, this shareholder can verify that her share was created in the correct way by checking the

$$g^{f(i)} = g^s \cdot g^{r_1 \cdot i} \cdot g^{r_2 \cdot i^2} \cdot \dots \cdot g^{r_{t-1} \cdot i^{t-1}} \bmod p.$$

In the reconstruction phase, each shareholder forwards the proof of the dealer such that only correct generated shares are used to reconstruct the secret.

Encryption Schemes

The motivation behind encryption schemes is to encode confidential messages in a way that the code can be transmitted over insecure channels to the intended reader of the message such that this person afterwards can decode the received code to obtain the confidential message.

Specification

Formally, an encryption scheme is a triple of algorithms (G, E, D), where G is a key generation algorithm, E is the encryption algorithm, and D the corresponding decryption algorithm. Encryption schemes can be *asymmetric* and *symmetric*: In the symmetric case, encryption key e and decryption key d are equal and therefore not known to the public, while for asymmetric encryption schemes $e \neq d$ and e is known to the public. Asymmetric encryption schemes can be further classified into *deterministic* and *probabilistic* asymmetric encryption schemes: deterministic schemes map identical messages to identical ciphertexts, as opposed to probabilistic encryption schemes that integrate randomness into the encryption procedure such that two encryptions of identical messages lead to distinct ciphertexts. In the remainder of this chapter we will denote ciphertexts of a message m encrypted under key k by $\{m\}_k^r$, where r denotes the optional randomness.

There exist a large number of encryption schemes, among which the most important symmetric schemes are *DES* (Data Encryption Standard) and *AES* (Advanced Encryption Standard). The first asymmetric and one the most influential deterministic asymmetric encryption scheme is *RSA* (Rivest, Shamir, & Adleman, 1978), and well-established probabilistic encryption asymmetric schemes are *ElGamal* (ElGamal, 1985) and *Paillier* (Paillier, 1999). In the remainder of this chapter, we focus on asymmetric encryption schemes as they build the basis of most electronic voting systems. A wide range of security notions expresses the security of asymmetric encryption schemes, among which the most important are *Indistinguishability under chosen-plaintext attack* (IND-CPA), *Indistinguishability under non-adaptive chosen ciphertext attack* (IND-CCA), and *Inditinguishability under adaptive chosen ciphertext attack* (IND-CCA2).

ElGamal Encryption Scheme

In this section we outline the ElGamal encryption scheme introduced in (ElGamal, 1985). This scheme turns out to be of value for electronic voting system due to its important homomorphic properties. Homomorphic cryptosystems allow the functional operations on plaintexts that result in a different functional operation on the corresponding ciphertext. Given two algebraic groups $(P,\oplus)$ and$(C,\otimes)$, then ϕ is a homomorphic mapping between groups $(P,\oplus)$ and $(C,\otimes)$ if for all $p_1, p_2 \in P$, it follows that

$$\phi(p_1 \oplus p_2) = \phi(p_1) \otimes \phi(p_2).$$

As outlined in the following, the homomorphic character of the ElGamal cryptosystems allow to implement a number of operation, such as the re-encryption of ciphertexts.

Key Generation

The key generation algorithm outputs a large prime p, a generator g for the multiplicative group Z_p^*. Furthermore, the algorithm outputs a random number $x \leftarrow \{2, \dots, p-2\}$ as private key and $(g, p, y = g^x \pmod p)$ as public key.

Joint Feldman Distributed Key Generation

We present an adaptation (Gennaro et al., 1999) of the distributed key generation scheme introduced in (Feldman, 1987). Goal of this scheme is to establish a joint public key such that the corresponding secret key is not known to anybody.

1. Participant i generates a polynomial of degree t over Z_q,

$$p_i(x) = a_{i,0} + a_{i,1}x + \cdots + a_{i,t}x^t,$$

where $a_{i,0}$ denotes the shared secret. For each participant j, participant i then computes $x_{i,j} = p_i(j)$ and provides j with that value. Furthermore, i commits on the generated polynomial p_i by publishing the values $X_{i,k} = g^{a_{i,k}}$ for all $0 \leq k \leq t$.

2. Each participant j verifies the shares obtained from all other participants by checking if equation

$$g^{x_{i,j}} = \prod_{k=0}^{t} X_{i,k}^{j_k} \bmod p$$

is satisfied. If this equation holds, j accepts, otherwise j publishes a complaint about i. If i is accused by more than t participants or if i does demonstrably not follow the protocol, i is excluded and $a_{i,0}$ is set to 0, while $X_{i,0}$ is set to 1.

3. The public value is computed by $y = g^a \cdot \prod_{i=1}^{n} X_{i,0} \bmod p$, while the secret value can be computed as $x = a + \sum_{i=1}^{n} x_{i,0} \bmod p$. The voter thereafter is able to compute the secret value if at least t out of n tellers behaved properly.

Encryption

Given a public key (g, p, y), a message $m \leftarrow \{0, \dots, p-1\}$ is encrypted with randomness $r \leftarrow \{2, \dots, p-2\}$ in the following way:

$$(c_1, c_2) = (g^r, m \cdot y^r) \bmod p$$

Decryption

Given a ciphertext (c_1, c_2) encrypted under public key (g, p, y), message m is reconstructed as follows:

$$m = c_2 \cdot c_1^{-x}$$

Homomorphic Property

The ElGamal encryption scheme satisfies an important property for electronic voting systems, namely it is homomorphic. Given two ElGamal ciphertexts $c_i = (g^r, m_1 \cdot y^r)$ and $c_j = (g^s, m_2 \cdot y^s)$ for messages m_1, m_2, it holds that $c_i \cdot c_j$ is a valid ciphertext of message $m_1 \cdot m_2$ as shown below.

$$c = c_i \cdot c_j = (g^r, m_1 \cdot y^r) \cdot (g^s, m_2 \cdot y^s) = (g^{r+s}, m_1 \cdot m_2 \cdot y^{r+s}) \bmod p$$

For electronic voting, it might be more useful to add messages rather than multiplying them. Therefore, the ElGamal encryption scheme has been extended towards additive homomorphism. The resulting scheme is called Exponential ElGamal (Cramer, Gennaro, Schoenmakers, 1997) and ciphertexts consequently have the following form:

$$(c_1, c_2) = (g^r, g^m \cdot y^r) \bmod p$$

It can easily be seen that the multiplication of individual ciphertexts results in the addition of the underlying plaintexts.

$$c = c_i \cdot c_j = (g^r, g^{m_1} \cdot y^r) \cdot (g^s, g^{m_2} \cdot y^s) = (g^{r+s}, g^{m_1+m_2} \cdot y^{r+s}) \bmod p$$

It should be noted that decryption of this ciphertext does not immediately results in m, but rather in g^m. Finally, the discrete logarithm of $g^{m_1+m_2}$ must be computed, which is only feasible for small exponents.

Re-encryption

Given a ciphertext $(c_1, c_2) = (g^r, m \cdot y^r) \bmod p$ encrypted under public key (p, g, y), this ciphertext can be re-encrpyted using randomness $s \leftarrow \{2, \dots, p-2\}$ in the following way:

$$(c_1', c_2') = (g^r \cdot g^s, m \cdot y^r \cdot y^s) \bmod p$$

The concept of re-encryption is extended to a set of ciphertexts encrypted under the same public key in straight-forward manner.

Distributed Decryption

So far, the concept of distributed key generation has been abstract. The concept proves however to be of great importance to distributed decryption. In distributed decryption, a ciphertext is partially decrypted by participants such that the partial decryption can be used to reconstruct the plaintext based on the Lagrange interpolation. Let an ElGamal ciphertext $c = (c_1, c_2)$ be given. Throughout the decryption phase, voter v_i computes her partial decryption

$$c_1(i) = c_1^{x_i}$$

and publishes a proof showing that

$$\log_{c_1} c_1(i) = x_i = \log_g y_i$$

If the voter's proof does not convince the majority of voters, they decide to reconstruct her private credential share in a distributed way relying on the Lagrange interpolation of the committed shares of the private key shares of voter v_i. The honest participants are capable of reconstructing x_i and hence $c_1(i) = c_1^{x_i}$.

Once, all voters' partial decryptions $c_1(i)$ are available, the plaintext is reconstructed by

$$m = \frac{c_2}{\prod_{i=1}^{n} c_1(i)}$$

Digital Signatures

The goal of signature schemes is to ensure the integrity and authenticity of messages with respect to the sender as well as non-repudiation.

Specification

A signature scheme is a triple of algorithms (G, S, V), where G is a key generation algorithm, S is the signing algorithm, and V the verification algorithm. The most significant security properties of digital signature schemes are *universal unforgeability* (UU), *selective unforgeability* (SU), and *existential unforgeability* (EU).

RSA Signature

Key Generation: Given two large primes p, q, two values $n = p \cdot q$ and $\varphi(n) = (p-1) \cdot (q-1)$ are computed. A value e with $1 < e < \varphi(n)$ co-prime to $\varphi(n)$ is randomly chosen and d is determined such that

$$e \cdot d \equiv 1 \bmod \varphi(n).$$

The verification key is (e, n), the signing key is d.

Signing: Given the signing key d, a message $m < n$ is signed according to the following equation:

$$s = m^d \bmod n$$

Verification: Given a verification key (e, n), signature s on message m is valid if the following equation holds:

$$s^e = m \bmod n$$

RSA Blind Signature

The RSA blind signature scheme has been invented in (Chaum 1981) and extends the standard RSA signature.

Blinding: The blinder randomly chooses a blinding factor $k \leftarrow Z_n^*$, blinds her message m and sends the corresponding value

$$b = H(m) \cdot k^e \bmod n.$$

to the signer.

Signing: The signer signs this value with her public key and sends the corresponding value s'

$$s' = b^d = (H(m) \cdot k^e)^d = (H(m))^d \cdot k^{ed} = \big(H(m)\big)^d \cdot k \ (\bmod n).$$

back to the blinder.

Unblinding: The blinder removes the blinding factor

$$s = \frac{s'}{k} = \frac{\big(H(m)\big)^d \cdot k}{k} = \big(H(m)\big)^d \ (\bmod n)$$

and obtains the signer's signature on her message m. Without further authentication step, the blinder can publish the message and the signature. Note that in the blinding phase, message m must be hashed in order to avoid exploits of RSA's malleability, i.e., a malicious blinder could obtain signatures m_1^d, m_2^d and deduce a new valid signature for $m_1 * m_2$ due to the fact that $(m_1 * m_2)^d = \big(m_1^d * m_2^d\big)$.

Zero-Knowledge Proof Systems

Zero-knowledge (ZK) proof systems are the cryptographic tool to prove the validity of statements without revealing anything beyond the validity of this statement.

Specification

A ZK proof system is given by a tuple of algorithms (P, V), where P is the prover of statements and V is the verifier of these statements. A ZK proof system for given language L satisfies three properties: 1) each valid statement can be proven (completeness), 2) no invalid statements can be proven (soundness), a malicious verifier does not learn anything beyond the validity of the statement (zero-knowledge). We will outline one prominent ZK proofs used in electronic voting systems, namely proof of knowledge of discrete logarithm, which can be used to exclude replay attacks in distributed key generation. There exist numerous further specific ZK proofs, e.g., designated-verifier proofs, proof of equality of discrete logarithms, 1-out-of-L encryption proofs, disjunctive proof of equality between discrete logarithms. We refer the interested reader to (Smith, 2005) for detailed information.

Proof of Knowledge of Discrete Logarithm

In (Schnorr, 1989), Schnorr invented a protocol to prove the knowledge of discrete logarithm. Given basis $g \leftarrow Z_p$, value $y \leftarrow Z_p$, the prover wants prove that she knows l such that $y = g^l$ where g and y are publicly known. The protocol is summarized as follows:

1. The prover randomly draws $r \leftarrow Z_p$ and outputs $a = g^r$
2. The verifier randomly draws $c \leftarrow Z_p$ and outputs c
3. The prover computes $z = r + l \cdot c$ and outputs z
4. The verifier checks if $g^z = a \cdot y^c$

Benaloh Challenge

In (Benaloh, 2006), Benaloh invented a concept to prove the integrity of encryptions in ZK proof manner. Assume a user intends to encrypt message m with a public encryption key pk using the ElGamal encryption scheme in an arbitrary system. Then, in accordance to the encryption algorithm, the system draws randomly $r \leftarrow \{2, \dots, p-2\}$ and computes

$$(c_1, c_2) = (g^r, m \cdot y^r) \bmod p.$$

The question arises how the user can be sure that the system encrypted the right value, anyway the output will be indistinguishable by definition for all input values. Benaloh proposed the following procedure: After encrypting m, the system commits on the encryption process by providing the user with $H\big((c_1, c_2)\big)$. The user thereafter (unpredictably) decides if she audits or accepts the encryption process of the device. If she decides to audit the process, the device returns the randomness r. The user can verify the correct encryption by computing $(c_1', c_2') = (g^r, m \cdot y^r) \bmod p$ locally or with the help of an external institution and checks if $H\big((c_1, c_2)\big) = H\big((c_1', c_2')\big)$. After the verification process, the voter has to re-run the entire encryption process. If the user at some point decides to obtain the ciphertext, the system provides the voter with (c_1, c_2) and a signature on it.

Electronic Voting System Properties

The conduction of elections is generally bound to legal constraints. For instance, the German Constitution prescribes the implementation of the six election principles *universal*, *direct*, *free*, *equal*, *secret* elections, as well as the *public nature* of the election. The principles must be refined into more concrete technical properties in order to apply them to voting technology. This has been done in an interdisciplinary dialogue. We identified 17 technical properties. Some can be directly addressed by the SnC technique, namely the properties secrecy, fairness, integrity, and verifiability, while others can only be evaluated on the basis of SnC techniques enriched with identification and authentication mechanisms; these are eligibility and uniqueness. Further technical properties can only be evaluated based on the fully implemented and organizationally running system, like usability and system availability. Correspondingly, the focus of this work is on secrecy, fairness, integrity, and verifiability.

Note, in the following analysis we do not consider how definite the relation between a voter and her cast vote is as this relation mainly depends on the form of identification and authentication, namely a voter who authenticates via password may easily forward her password, while a voter authenticating via her national ID card may not do so. On the other hand, with regard to secrecy, we merely consider any relation between a voter and her vote as crucial, independent of the precise identification and authentication mechanisms, e.g. also considering the voter's IP address.

Secrecy, Fairness, Integrity, and Verifiability

Secrecy and fairness are closely related and stem from an election principle enshrined in many national and international constitutions, namely the *secret election* principle. Amongst other principles, integrity is derived from the *universal* and *equal election* principles. Verifiability on the other hand implements the *public nature* principle on a technical level. Even though the public nature principle is not embodied in all democratic states' constitutions, it turns out to be of central importance for electronic voting systems for two reasons: first, electronic voting systems face the dangers of large-scale manipulations (Mercuri, 2002); second, it might increase trust in the voting system. Our interdisciplinary project work led us to the following definitions:

Secrecy: For each voter v that cast a vote for an arbitrary candidate c, it holds that the adversary cannot get more evidence about the fact if the voter selected c or any other selection c' as he can get from the final tally. Note that a selection depends on the electoral systems and might include the

voting for multiple candidates, for instance in ranked voting methods. In this paper, we focus on *single-candidate elections.*

Fairness: The adversary cannot obtain any evidence about any cast intention before the end of the election.

Integrity: *Integrity* is composed of three sub-properties:

Cast-as-intended: The voter's cast vote corresponds to her intention. Note that votes are usually prepared before being cast to ensure secrecy by techniques like.

Stored-as-cast: The voter's cast vote is stored for tabulation the way she cast it.

Tallied-as-stored: All votes have been tallied the way they were stored.

Verifiability: In analogy to the integrity definition, *verifiability* is composed of three sub-properties:

Cast-as-intended: The voter can individually verify the proof that her vote has been cast the way she intended to cast it.

Stored-as-cast: The voter can individually verify the proof that her vote has been stored for tabulation the way she cast it.

Tallied-as-stored: Anybody can verify the proof that all votes have been tallied the way they were stored.

The proofs mentioned above must be sound; hence, there must be no possibility for the adversary to generate proofs for wrong statements that pass the verification step. Note, thereby, we define verifiability as the strongest form of integrity. In the literature, verifiability is often also referred to as *end-to-end verifiability*, if all three verifiability sub-properties are given.

In practice, the average voter is not able to verify these proofs manually as these proofs are usually based on complicated cryptographic primitives. Therefore, she needs to rely on some support. Correspondingly, we define *verify* not by voters being personally able to verify proofs manually, but rather they can use arbitrary hard-/software to verify proofs. Verifiability is only given if the hardware is provided from different manufacturers and the software from different developers because then voters can choose which manufacturers and which developers to trust and use their hardware and software respectively, where software includes the operating system. We give some examples for a better understanding in the later analysis: Consider the Benaloh challenge (Adida, 2006) implementation in the Helios system (Adida, 2008): In theory, the system allows to output auditing data for external auditing of the encryption process. The implementation as used in Helios embodies the possibility to forward auditing data from the JavaScript to external auditors. Cast-as-intended verifiability is not ensured in this implementation as voters conduct the verification process within their voting environment, i.e., the environment that they use to cast a vote. This voting environment covers the hardware as well as the operating system and browser used by the voter. One way of ensuring cast-as-intended integrity without adversarial assumptions has been indicated in (Karayumak et al., 2011). The authors propose to outsource the auditing process via QR codes to an external device, e.g., smartphone, in order to achieve cast-as-intended verifiability.

Further Security Properties

Electronic voting systems have to ensure more security properties than secrecy, fairness, integrity, and verifiability. In addition, they must ensure that only eligible voters can cast valid votes (*eligibility*) and each eligible voter can cast exactly one valid vote (*uniqueness*). As these can only be ensured by combining SnC techniques with corresponding identification and authentication mechanisms, these two properties are not considered in the main analysis. However, as the different SnC techniques have to be compatible with the different identification and authentication mechanisms, we add a brief general discussion of the SnC techniques' adequacy with respect to eligibility and uniqueness.

Evaluation Criteria

Throughout this section, we specify the evaluation criteria for SnC techniques used in the following analysis. As first evaluation criterion, we specify the underlying *security model* of the SnC techniques thereby measuring the strength of techniques with respect to secrecy, fairness, integrity, and verifiability. Our second class of evaluation criteria covers the adequacy of security and cryptographic techniques to address *further properties*, including *further security properties*, as well as *costs*.

Security Model for Security Properties

The criterion security model consists of two distinct sub-criteria addressing the secrecy, fairness, integrity, and verifiability property. The first criterion determines the adversary model against which a technique can maintain secrecy, fairness, and integrity. The second criterion analyses the degree of verifiability provided by the corresponding technique.

Adversary Model

We specify adversary models by a capability-based approach as proposed in (Amenaza Technologies Limited, 2005). In the capability-based approach, the SnC technique is related to a mapping between security properties and assumptions (exclusion of adversarial capabilities) under which those properties can be ensured. Consequently, the adversary is defined by his capabilities at disposal.

In the next step, adversarial capabilities are determined that allow composing adversary models. The adversarial capabilities are based on a literature review and the composition of several existing approaches defining security models (Dolev & Yao, 1981; Langer, 2010; Carlos et al., 2013). We classify the identified adversarial capabilities in four subclasses, namely communication-based, corruption-based, computational capabilities, and timing capabilities. In the following paragraphs, we introduce the different categories and the corresponding adversarial capabilities.

Communication-based Capabilities

Originally, the Dolev-Yao communication and adversary model (Dolev & Yao, 1981) considered an adversary controlling the network between abstract entities. In a recent work (Carlos et al., 2013), Carlos et al. extended the Dolev-Yao communication model to fit *security ceremonies* (such as electronic voting) thereby distinguishing between human entities and computer systems. In the voting scenario, an adversary might control network channels between computer systems (e.g. the Internet), network channels between human entities (e.g. postal mail), or network channels between human entities and computer systems (e.g. the voter reading content on the display or interacting with her computer systems via typing and moving the mouse). In accordance to the extended Dolev-Yao model (Carlos et al., 2013), the following communication-based capabilities are specified:

1. The adversary can drop messages from the network channel.
2. The adversary can read messages on the network channel.
3. The adversary can inject messages on the network channel.

In the voting scenario, it might be enough for the adversary to determine the sender of a specific message in order to violate a voting system property. To address this issue adequately, Langer (Langer, 2010) specifies the following communication-based capabilities:

4. The adversary can recognize the sender of messages on the network channel.
5. The adversary can notice the usage of a network channel.

Corruption-based Capabilities

The first corruption-based capability models adversaries capable of controlling single human entities involved in the election process. A human entity corrupted by the adversary is completely under adversarial control. These human entities might for instance be tallying authorities or key

shareholders but not voters, as they are handled separately in the following. Therefore, the following corruption-based capability is specified:

6. The adversary can corrupt a human entity.

As opposed to other human entities involved in the election process, we acknowledge that voters generally try to defend against adversarial attacks by cheating the adversary. This stems from the fact that without any adversarial action, voters do not have any motive not to vote according to their real intention. Therefore, adversaries cannot completely control the voter. Nevertheless, the adversary might try to influence voters by a variety of approaches in order to achieve his goal. Therefore, with respect to electronic voting, Langer (2010) extended these capabilities by new network channels that allow expressing more fine-grained security models such as indirect or bidirectional network channels between the voter and the adversary.

The adversary might convince the voter into proving her vote to the adversary in order to experience certain benefits. This capability stems from attacks in which the voter intends to forward objects obtained throughout the voting process in order to prove the way she voted (refer for instance to (Adida & Neff, 2009)). Therefore, Langer (2010) specifies the following corruption-based capability:

7. The adversary can obtain objects from a voter.

Adversaries might also be capable of sending objects to the voter. Objects an adversary might send in advance to the voting phase are instructions as abstaining from the election, signatures for Italian attacks (refer for instance to (Teague, Ramchen, & Naish, 2008)) or a random value in order to launch a randomization attack (refer for instance to (Ryan & Teague, 2009)). This capability models the adversary's power to blackmail or convince voters in advance to the voting phase into voting according to the adversary's intention. Therefore, Langer (2010) specifies the following corruption-based capability:

8. The adversary can send objects to a voter.

Having made explicit all the capabilities with respect to human entities, we now consider the second type of entities in our voting systems: computer systems. Computer systems are often not directly controlled by human entities and should therefore be separated from authorities or voters. Therefore, we introduce the following corruption-based capability:

9. The adversary can corrupt a computer system.

Computational Capabilities

Several works, e.g. (Sandler & Wallach, 2008), acknowledge the weaknesses of cryptosystems as they might be broken within few years. Voting systems might therefore differ with respect to the adversarial computational power against which they are able to defend certain security properties. Therefore, the following capability is specified:

10. The adversary is computationally unrestricted.

Note, unless otherwise stated, we restrict our attention in this chapter to computational rather than information-theoretic secrecy. To us, this seems most natural as information theoretic security generally comes along with unrealistic assumptions. We refer the interested reader to (Moran & Naor, 2007) for information-theoretic secrecy in electronic voting systems.

Timing Capabilities

Moreover, the adversary might possess the above listed capabilities only throughout a restricted time span. This restriction is for instance motivated by the facts that the adversary might not observe all voters simultaneously casting their votes (i.e. the network channels between voters and their computer systems), and might not continuously have access to channels between human entities (for instance refer to (Carlos et al., 2013)). Therefore, we specify the following timing capability:

11. The adversary has capability [1 - 10] during a specified period of time.

Degree of Verifiability

We evaluate SnC techniques with respect to the degree of verifiability they can assure. As a result of the integrity analysis, one can derive how many integrity sub-properties are ensured without posing assumptions on the adversarial capabilities. In the following analysis, all integrity sub-properties are of equal importance. We therefore define the second security model criterion degree of verifiability by a ratio of the form zero/one/two/three out of three. Note that we restrict our attention to *computational* rather than *information-theoretic verifiability*. We justify this constraint by the fact that verification usually takes place during or immediately after the voting phase, thus time is very restricted. We furthermore deliberately assume that the adversary cannot alter data written on the bulletin board, i.e., data which has been written on the bulletin board cannot be undetectably manipulated. This assumption is justified by the fact that the bulletin board is under the continuous supervision of the general public.

Criteria for further Properties

We also briefly study the relation between SnC techniques with properties that go beyond secrecy, fairness, integrity, and verifiability. First, SnC techniques are discussed with respect to their adequacy to implement eligibility and uniqueness. Second, we postulate that electronic voting should not higher the burden for democratic processes, but should rather be better competitive with conventional voting systems. We therefore evaluate SnC techniques and their designated implementation according to the criteria *cost*, covering administrative, architectural, and computational resources. These costs highly depend on the concrete implementation including the identification and authentication in place. Therefore, we leave this aspect for future work.

Analysis of Security and Cryptographic Techniques

The goal of this section is to review established and well known SnC techniques in electronic voting systems and evaluate them with respect to the defined evaluation criteria. In the first part of this section, we provide the reader with some background information about the structure of this section and the selection process. Thereafter, we describe and analyze the selected SnC techniques.

Background

We structure the SnC techniques according to the secrecy technique in place and thus similar to (Volkamer, 2009) according to the phase (pre-voting, voting, post-voting) in which the link between a voter and her vote is broken. Thus, we first consider simple and more complex code voting in the pre-voting phase. Then, online randomized authentication token and blind signature approaches are discussed as the most popular representatives for the second phase. As representatives of the third phase we consider shuffles and homomorphic cryptosystems.

Furthermore, we extend these standard techniques by the two concrete voting protocols – Civitas (Clarkson, Chong, & Myers, 2008) and Pretty Good Democracy (Ryan & Teague, 2009) – as each of them combines two different of the previously mentioned techniques.

Accordingly, we start describing the SnC techniques from a secrecy perspective, and if possible enhance the description towards verifiability.

For each described SnC technique, we identify the security model underlying secrecy, fairness, integrity, and verifiability. Note, due to the lack of space, we do not outline the complete security model but merely restrict our attention to assumptions which require the smallest number of distinct adversarial capabilities. We conclude the analysis with the analysis of further properties.

Approaches ensuring Secrecy in the Pre-Voting Phase

A technique is assigned to the pre-voting phase, if the voter's interaction with the electronic voting system is never associated with her identity; hence, the relation between a voter and her cast vote is broken in advance to the voting phase. The only representatives in this group of techniques are

the various code voting schemes which are discussed in this section. The idea of code voting goes back to the work of Chaum (Chaum, 2001).

Description

In the pre-voting phase, the registration authority prepares unique codebooks for all eligible voters: a codebook contains the codebook ID and a three-column table, where each *candidate* has a *voting code* and an *acknowledge code* assigned. After the generation of these books, the registration authority randomly assigns codebooks to voters and provides the tallying authority all issued codebooks. The voter must not receive her codebook over her voting environment (but for instance via postal mail). Thereby, the link between a voter and her vote is already broken in the pre-voting phase. In the voting phase, the voter casts her vote by sending the codebook ID and the voting code next to the preferred candidate to the tallying authority. The tallying authority re-interprets the code, identifies the chosen candidate and stores a vote for that candidate. Thereafter, the tallying authority returns the corresponding acknowledge code to the voter. Thereby, the voter gets assurance that her voting code was not manipulated or dropped by her system or on the communication channel. A voter can use her codebook ID to cast several votes giving her the possibility of vote updating. In this case, the voter's old vote is replaced by her new vote. In the post-voting phase, the tallying authority publishes all interpreted candidates on the bulletin board. This allows any observer to tally the result.

In order to improve the degree of verifiability, VeryVote (Joaquim, Ribeiro, & Ferreira, 2009) has been proposed. That scheme integrates the idea of code voting with MarkPledge (Neff, 2004) codes. A generic election authority generates codebooks for each voter in which each candidate has a unique voting code assigned. Furthermore, each codebook has a special acknowledge code, a so called MarkPledge code which is outlined in the following. In advance to the election, for each voter, the authority generates $n-1$ probabilistic bit encryptions of 0 and one bit encryption of 1, denoted by $BitEnc(0)$ and $BitEnc(1)$ respectively. The authority commits on them by publishing them on the bulletin board together with the voter's identity. Afterwards, a public challenge $srev$ (which is used to derive individual challenges in the voting phase) is distributively computed. Due to encoding properties $BitEnc(1)$ is partially opened independently of the challenge, while the partial opening of $BitEnc(0)$ depends on the challenge. The static, partial opening of the $BitEnc(1)$ encryption is referred to as MarkPledge code. After the voter cast her voting code to the election authority, her code is interpreted. The authority assigns the $BitEnc(1)$ to the chosen candidate and $BitEnc(0)$ randomly to the other candidates. The combination of candidates with the voter respective bit encryptions corresponds to the voter's encrypted ballot. Depending on the individual challenge, the authority reveals partial randomness used to generate the $BitEnc(0)$ values and the $BitEnc(1)$ value within the ballot. The authority publishes the partial decryptions of the ballot (the acknowledge codes), which is exactly the voter's receipt. The revealing of partial randomness values does not interfere with the secrecy property, which is discussed in (Joaquim, Ribeiro, & Ferreira, 2009). The voter can individually verify that on her public receipt, her acknowledge (MarkPledge) code appears next to her selected candidate. Furthermore, any observer can verify that the published randomness values correspond to the challenge and that the $BitEnc(x)$ encryptions correspond to the claimed $ack(x)$ codes, among which there is the MarkPledge code, only known to the authority and the voter. After the voting phase, published ballots are anonymized and ballots are decrypted by a set of trustees.

While this improvement regarding verifiability is rather complicated, one can think of the following simple straightforward improvement: After the voter cast her voting code, the corresponding acknowledge code is also published on a bulletin board. After the voting phase, the authority assigns each acknowledge code with the corresponding candidate. The following analysis shows that both improvements result in the same security model.

Security Model

Adversary Model: As opposed to most other SnC techniques, *secrecy* in the code voting approach does neither assume the voting environment nor the standard communication channel between the voter and the tallying authority to be trustworthy. All information the adversary might obtain controlling the environment or this communication channel cannot be mapped to the

corresponding selection. On the other hand side, it must be assumed that the adversary cannot read the channel between the registration authority and the voter (C2). However, the registration and tallying authority must be trusted not to collaborate because then the registration authority would keep track of which codebook was sent to which voter while the tallying authority knows which candidate was selected from which codebook. Note that, in the improved version, the adversary only needs to control the registration authority (C6). For both approaches - the straightforward improvement, and the VeryVote approach - the election/tallying authority is generic (the authority's duties might be separated) such that a final adversary model cannot be assessed. Nonetheless, both approaches rely on the fact that the voter must not be under adversarial control otherwise she can forward her codebook to the adversary. Thereby, she maintains the link between her identity and her vote due to the published acknowledge code. Hence, the adversary must not obtain information from the voter (C7).

Fairness in the code voting approach relies on the proper behavior of the generic election authority (C6) because this authority knows the relation between codes and candidates and furthermore receives voters' selected codes allowing this authority to compute intermediate results. After a voter cast her voting code, the corresponding acknowledge code is published on the bulletin board. If the voter forwards her voting material to the adversary (C7), the adversary can learn the voter's selection and compute an intermediate result.

Due to the fact that in the post-voting phase, acknowledge codes are publicly mapped to candidates, *cast-as-intended integrity* is not built upon any assumptions. With respect to *stored-as-cast integrity*, a malicious registration authority (C6) could assign identical codebooks to voters that predictably make the same selection. In that case, acknowledge codes are published once and the tallying authority (C6) could discard all votes with the same acknowledge code; correspondingly no additional candidate is stored. Finally, *tallied-as-stored integrity* does not rely on any adversarial assumptions.

Degree of Verifiability: The integrity analysis shows that *cast-as-intended* and *tallied-as-stored* are verifiable. Consequently, the degree of verifiability equals two out of three.

Further Properties and further Criteria

Eligibility and uniqueness depend on the authorities' proper behavior. Voters might forward their codebook to the adversary thereby violating either eligibility or uniqueness because the adversary has all information necessary to cast a vote. Sticking to conventional voting systems, the channel between the registration authority and the voter might be implemented by postal mail, which is the main pillar of the security assurance; hence in the pre-voting phase, each voter receives a letter containing a random codebook. This might result in significant administrative *costs*.

Approaches ensuring Secrecy in the Voting Phase

A technique is assigned to the voting phase, if the link between a voter and her vote is broken as part of the interaction with the electronic voting system. Generally, those techniques involve several voter interactions with the system. Representatives of this group are online randomized authentication tokens and blind signature approaches.

Online Randomized Authentication Token

The concept of randomized authentication tokens might be interpreted as a separation of duties between a registration and tallying authority. Randomized authentication tokens are used in the POLYAS voting system, which is used for the GI (German Computer Science Society) elections (Olembo, Kahlert, Neumann, & Volkamer, 2012).

Description

In the pre-voting phase, the registration authority generates random tokens for all eligible voters. In the voting phase, once an eligible voter authenticates towards the registration authority throughout the voting phase, the voter's ID is marked in the electoral roll and a random token is

returned to the voter. This random token is forwarded to the tallying authority. Note that the entire registration process might also be offline; then the approach would be a pre-voting approach and tokens could be returned randomly to the voter similar to the code voting approach. After the voter received her randomized authentication token, she can continue with making her selection, and subsequently cast her vote. Therefore, the voter prepares a tuple containing her token and her selection and casts this tuple to the tallying authority. The server accepts the vote if the associated token has been forwarded by the registration authority during registration. The voter might use her token several times, in order to update her vote. In the post-voting phase, tallying authority separates the tokens from the votes and publishes the votes on the bulletin board. The registration authority publishes the issued tokens on the bulletin board.

One might consider the following improvement: In order to cast her vote v, the voter uniquely identifies her vote by generating a large random number r and casting the tuple

$$(v||r)$$

to the tallying authority, which is published in the post-voting phase.

Security Model

Adversary Model: If the adversary does not know the link between a voter and the assigned token, the adversary might use the IP address to infer the sender's identity and to establish a link between the voter and her vote, thereby violating *secrecy*. Hence, it is assumed that the adversary cannot determine the origin of messages on the channel between voters and the tallying authority (C4). Secrecy in the randomized authentication token approach relies on the distribution of trust among the registration and the tallying authority. If both authorities collaborate, then the link between a voter and her vote can be established. This leads to the following assumption that the adversary cannot control the registration and the tallying authority simultaneously (C6). For the sake of verifiability, votes are unique. Therefore, it must be assumed that the voter is not under adversarial control before the post-voting phase (C7: before post-voting). Furthermore, it must be assumed that the adversary does not control the voting environment (C9).

The technique of randomized authentication tokens ensures *fairness* under the assumption that the tallying authority does not publish votes unless the election is declared terminated (C6). If the adversary is capable of controlling the voter's voting environment (C9), it can learn this voter's selection. Finally, if the adversary can read the channel between the voter and the tallying authority (C2), this adversary can compute an intermediate result.

Due to the fact that published votes are human-readable, *cast-as-intended integrity* is implicitly given. Analogous to the previous techniques, *stored-as-cast integrity* relies on several assumptions as the following attack indicates: If the adversary controls several voters' machines (C9) and the tallying authority (C6), voters casting identical votes could be provided with identical randomness by their machines. If the authority additionally only stores one vote from these voters and discards other, stored-as-cast integrity is violated. This attack has been defined by Küsters and Truderung (2012) as clash attack. Because any public observer can re-compute the final result from the stored votes ensuring, *tallied-as-stored integrity* does not rely on any adversarial assumptions.

Degree of Verifiability: The above analysis shows that *cast-as-intended* and *tallied-as-stored verifiability* are given, while stored-as-cast integrity relies on the absence of several adversarial capabilities; hence, two out of three sub-properties are verifiable.

Further Properties and further Criteria

There are three ways to violate *eligibility* or *uniqueness*: If voters abstain from the election after receiving their token, the tallying authority might use these tokens to vote on their behalf. Second, the registration authority might provide tokens to ineligible voters. Third, voters might forward their tokens to ineligible voters. The concept does not rely on complex cryptography and neither voters nor authorities need to conduct complex computations. Even low-resource devices with authentication capabilities might be used to store tokens and release them in the voting phase. Only

three servers and administration staff to these servers will be involved which can be provided at low *costs*.

Blind Signatures

Originally, blind signatures were introduced to implement digital cash (Chaum, 1981) and have later one been applied to electronic voting systems (Fujioka, Okamoto, & Ohta, 1992). Blind signatures are a specific form of digital signatures in which a signer signs a blinder's message without knowing the message's content after the blinder's successful authentication. Hence, the signer's signature on this message confirms the authentication process of the message's origin. Already in 2000, a simple implementation of blind signatures was used in smartcards to execute a student parliament election at the University of Osnabrück, Germany (Klink, 2006).

Description

Similar to code voting, the blind signature technique is based on a separation of duty approach with two authorities, a registration and a tallying authority. Blind signature based voting systems separate the voting phase into a registration and a vote-casting step.

According to (Fujioka, Okamoto, & Ohta, 1992), a voter makes her selection v_i, blinds this selection, sends the blinded vote to the registration authority, and authenticates towards the registration authority. The authority conducts the eligibility check and signs the blinded vote in case the voter is eligible. The signed blinded vote is returned to the voter. The voter unblinds her vote in such a way that the signature verifies for the unblinded vote; hence the voter obtains an officially signed vote. In the vote-casting step, the voter casts her signed vote to the tallying authority. The authority checks the validity of the signature and stores the vote if the verification succeeds. In the post-voting phase, the tallying authority publishes all valid data received from voters together with the result on the bulletin board.

In (Okamoto, 1996; Okamoto, 1997; Xia & Schneider, 2006), beside others, this voting protocol has been improved regarding verifiability. The system relies on several registration and tallying authorities. In the pre-voting phase, the voter draws a random value $a_i \leftarrow Z_q$ and calculates $h_i = g^{a_i}$. The voter makes her selection v_i and draws a random number r_i. She computes

$$m_i = g^{v_i} \cdot {h_i}^{r_i} \bmod p$$

Using a fixed randomness a_i, the voter can compute m_i in several ways by changing v_i, r_i. The only constraint for the voter is to satisfy equation

$$v_i + a_i \cdot r_i \equiv v_i' + a_i \cdot r_i' \bmod q.$$

The voter thereafter draws a blinding factor k and calculates

$$x_i = H(m_i || h_i) \cdot k^e$$

The voter signs her blinded value, authenticates and sends tuple $(sig(sk_i, x_i), x_i)$ to the authority. In accordance to the above description, the registration authority blindly signs the value x_i and returns it the voter. The voter in turns unblinds it and obtains

$$s_i = H(m_i || h_i)^d$$

In the voting phase, the voter posts $(m_i || h_i, s_i)$ on the bulletin board and sends (m_i, v_i, r_i) to the tallying authority. In the post-voting phase, the registration authority publishes the list of tuples $\big((sig(sk_1, x_1), x_1), \dots, (sig(sk_k, x_k), x_k)\big)$. The tallying authority publishes a randomized list of votes $(v_1, \dots, v_k)$ together with a *ZK proof* showing that for each v_i there is a m_i' on the bulletin board such that the authority knows r_i such that $m_i' = g^{v_i} h_i^{r_i}$ without revealing the relation between v_i and m_i'.

Security Model

Adversary Model: *Secrecy* in the blind signature approach relies on the anonymous transmission of data to the tallying authority (C4). In (Okamoto, 1996; Okamoto, 1997), the voter can

be under adversarial control, as the voter can generate receipts for different selections by switching v_i, r_i such that $m_i = g^{v_i} \cdot {h_i}^{r_i} \bmod p$ is satisfied. However, it must be assumed that the adversary does not control the voting environment (C9); otherwise the adversary can easily obtain a voter's selection and a_i and consequently detect fake receipts.

The tallying authority has access to intermediate results at any time; hence, *fairness* relies on the trustworthiness of that authority (C6). The adversary can also compute intermediate results if he is capable of reading the channel between the voter and the tallying authority (C2). Finally, a voting environment under adversarial control can release the selection(s) cast over that environment (C9).

After the voter unblinds her commitment, she can verify the validity of the authority's signature without restricting the adversary anyhow; hence, *cast-as-intended integrity* does not build upon environmental assumptions. The voter publishes her committed vote on the bulletin board. However, an attack similar to the code voting attack is possible: If the authority controls several voters' machines (C9), she could choose a_i, r_i identically for voters casting identical votes while all would refer to the same entry on the bulletin board without noticing. If additionally the bulletin board behaves dishonestly (C6 or C9), these voters would find $(m_i || h_i, s_i)$ on the bulletin board while only one vote is stored. *Stored-as-cast integrity* therefore depends on the absence of the listed adversarial capabilities. In the post-voting phase, the tallying authority publishes a list of votes and proves that each vote corresponds to exactly one de-committed vote stored on the bulletin board; this ensures *tallied-as-stored integrity* without any assumptions.

Degree of Verifiability: The above integrity analysis shows that *cast-as-intended* and *tallied-as-stored* are verifiable; hence, two out of three sub-properties are verifiable.

Further Properties and further Criteria

If voters abstain from the election, the registration authority can issue valid signatures for own commitments and consequently cast valid votes, thereby violating *eligibility* or *uniqueness*. In the simplest approach, blind signatures rely on a registration, a tallying authority, and a bulletin board. The process foresees the registration authority to sign a blinded item while the tallying authority verifies the signature in the voting phase; the voter's task is to blind and unblind her item. Hence the computational effort and administrative effort is at a low level for both authorities and the voter, and hence *costs* are moderately.

Approaches ensuring Secrecy in the Post-Voting Phase

The techniques discussed in this section have in common that there is a link between the voter and her encrypted vote; therefore, a voter usually posts her encrypted vote together with her ID or her pseudonym on the bulletin board. The two representatives of this class of SnCs are homomorphic cryptosystems and shuffle techniques.

Homomorphic Cryptosystems

Rather than decrypting individual votes, first the encrypted sum of all encrypted votes is computed and then this value is decrypted to determine the result. This is possible if an encryption schemes with additive homomorphic properties, such as Exponential ElGamal scheme (Cramer, Gennaro, Schoenmakers, 1997) or the Paillier scheme (Paillier, 1999) is in place. The homomorphic cryptosystem approach has been implemented in Helios 2.0 and has been used to conduct the President election at the Université catholique de Louvain. An experience report and analysis of the real-world use can be found in (Adida, Pereira, de Marneffe, & Quisquater, 2009).

Description

In the simplest case of referendum (Yes/No election), homomorphic encryption schemes can be implemented in a straightforward manner. First voter i makes her selection $v_i \in \{0,1\}$ and encrypts

her selection with the public key of the key trustees pk. In order to be convinced that her encrypted ballot contains the vote she intends to cast, the *Benaloh challenge* is in place. The voter thereafter binds her authentication data to her encrypted vote, e.g., by posting her name together with $\{v_i\}_{pk}^{r_i}$ on the bulletin board. The voter furthermore provides a proof that her vote is a valid vote in order to prevent malicious voters from over-voting (i.e., a proof showing that $v_i \in \{0,1\}$). The voter can convince herself that her vote was stored in an unaltered way on the bulletin board by checking if her name appears next to her encrypted vote and the corresponding proof.

In the post-voting phase, the public can calculate the encrypted result by multiplying the encrypted individual votes.

$$R = \{v_1\}_{pk}^{r_1} \cdot \ldots \cdot \{v_n\}_{pk}^{r_n}.$$

The result can be computed by decrypting the product R with the corresponding secret key; hence

$$r = D(sk, R).$$

Finally, the key trustees prove that they properly decrypted, i.e. that they decrypted the product of encrypted votes with the proper secret shares by generating a ZK proof of correct decryption based on a ZK proof of equality of discrete logarithms.

Security Model

Adversary Model: The *secrecy* of homomorphic cryptosystems is based upon the trustworthiness of the key trustees. Hence, under the assumptions that the threshold of key trustees behaves properly (C6), individual votes are not decrypted and secrecy therefore is ensured. In the voting phase, voters only receive encrypted votes and proofs for the purpose of verifiability. Even voters forwarding this data cannot break secrecy, because this information cannot be used to reconstruct the voter's selection. Furthermore, it must be assumed that the adversary does not control the voting environment (C9); otherwise the voter's selection might be forwarded to the adversary.

Under the assumption that the threshold set of key trustees is trustworthy (C6), homomorphic cryptosystems provide *fairness* because individual votes are not decrypted at any time. Furthermore, the voter's voting environment must not be under adversarial control (C9) in order not to release any voter's selection thereby providing intermediate results.

Without posing assumptions on the adversarial behavior, homomorphic cryptosystems ensure *cast-as-intended integrity* due to the Benaloh challenge, *stored-as-cast integrity* due to publishing their identifiable encrypted votes, and *tallied-as-stored integrity* due to the ZK proofs of correct decryption.

Degree of Verifiability: The integrity analysis shows that the homomorphic cryptosystem approach is *verifiable*.

Further Properties and further Criteria

In the homomorphic approach, *eligibility* and *uniqueness* strongly depend on the authentication data a voter assigns to her encrypted vote. Due to the fact that identification and authentication mechanisms are not considered in this work, eligibility and uniqueness are not evaluated for this approach. It turns out that homomorphic cryptosystems for electronic voting suffer a significant drawback, namely the computational effort. Rather than providing the sum $v_1 + \cdots + v_n$ of all votes, decryption of the Exponential ElGamal ciphertext yields $g^{v_1 + \cdots + v_n}$. This bottleneck might be overcome by the more efficient Paillier homomorphic cryptosystem. Furthermore, voters need to provide evidence that their cast votes encode valid votes. Implementing complex ZK proofs on low-resource devices might be problematic, hence the *cost*s when implementing this approach might be considered to be high.

Shuffles

In (Chaum, 1981), Chaum invented the technique of digitally shuffling messages in order to allow anonymous communication over insecure communication networks. Shuffles implement communication protocols among a set of *shuffle nodes* where each node receives a batch of incoming messages, shuffles them according to a secret and random permutation, modifies their appearance (this will be specified shortly), and forwards the anonymized messages to the next shuffle node.

Applied in electronic voting systems, shuffles are used to break the link between a voter and her encrypted vote before decrypting the vote. Two shuffle implementations are distinguished: decryption shuffles and re-encryption shuffles. Due to their impact on electronic voting systems, in this work we merely consider re-encryption shuffles (decryption shuffles are e.g. used in (Clarkson & Myers, 2005)). The first verifiable re-encryption shuffle was presented by (Sako & Kilian, 1995). Many techniques have been presented to make these shuffles efficiently verifiable, among which we refer the reader to (Wikström, D., 2005; Chase et al., 2012). These verifiable re-encryption shuffles rely on the use of malleable encryption schemes, i.e., schemes that allow re-encryption of ciphertexts without seeing and changing the contained plaintext. In (Jakobsson, Juels, & Rivest, 2002), the authors propose randomized partial checking as technique to improve the efficiency significantly, while at the same time reducing the degree of verifiability provided. The shuffle-based approach has been used in the Norwegian municipality elections in 2011 (Gjøsteen, 2010) while all shuffle nodes were provided by the same company (OSCE/ODIHR, 2012).

Description

Generally speaking, a voter encrypts her vote with the public key of the key trustees pk. The voter thereafter binds her authentication data to her encrypted vote, e.g., by posting her name together with $\{v_i\}_{pk}^{r_i^1}$ on the bulletin board. Analogous to the homomorphic approach, the voter can convince herself that her vote encoded in the ciphertext due to the Benaloh challenge. The voter can furthermore convince herself that her vote was stored in an unaltered way on the bulletin board. After the voting phase, encrypted votes are separated from the voter's ID and passed through a verifiable re-encryption shuffle; all data generated by each shuffle node is published on the bulletin board. The set of encrypted votes published by the last shuffle note is decrypted vote by vote by the threshold set of key trustees and published on the bulletin board. Afterwards these votes are tallied and the result is published as well. Here, each shuffle node *j* re-encrypts each voter's encrypted vote $\{v_i\}_{pk}^{r_i^{j-1}}$ with additional randomness $r_{j1} \dots, r_{jn}$ resulting in overall randomness $r_1^j \dots, r_n^j$, draws a random permutation ψ_j, and performs the following transition for all votes:

$$\phi\{v_1\}_{pk}^{r_1^{j-1}}, \dots, \phi\{v_n\}_{pk}^{r_n^{j-1}} \rightarrow \psi_j\left(\phi\{v_1\}_{pk}^{r_1^j}\right), \dots, \psi_j\left(\phi\{v_n\}_{pk}^{r_n^j}\right)$$

Each shuffle node has to prove that the correct processing of received ciphertexts. Therefore, each node proves that incoming ciphertexts contain the same plaintexts as the outgoing ciphertexts without revealing the plaintexts. One approach is based on the Chaum-Pedersen protocol (Chaum & Pedersen, 1992) which allows a shuffle node to prove that for all $i \in \{1, \dots, n\}$, $\psi_j\left(\phi\{v_i\}_{pk}^{r_i^j}\right)$, is a re-encryption of $\phi\{v_i\}_{pk}^{r_i^j}$ without revealing the relation. This proof can be implemented based on a 2×2 shuffle proof which is realized by OR-ing ZK proofs of equality of discrete logarithms as proposed in (Smith, 2005).

Security Model

Adversary Model: The *secrecy* relies on the trustworthiness of at least one shuffling node; hence, it must be assumed that not all shuffling are under adversarial control (C6). It must be assumed that the threshold set of key trustees is trustworthy (C6) and does not decrypt those encrypted votes that are published on the bulletin board together with the voters' names. Furthermore, a malicious voting environment would easily break secrecy by forwarding the voter's selection to the adversary or

by maliciously fixing the randomness used to encrypt the voter's selection. Hence, it is assumed that the adversary cannot control the voting environment (C9).

Similar to the homomorphic cryptosystem approach, *fairness* can be ensured under the assumption that the threshold set of key trustees is trustworthy (C6) and the voter's voting environment is trusted (C9).

In analogy to the homomorphic cryptosystem approach, without posing assumptions on the adversarial behavior, the shuffle-based approach ensures *cast-as-intended integrity* due to the Benaloh challenge, *stored-as-cast integrity* due to publishing their identifiable encrypted votes, and *tallied-as-stored integrity* due to the ZK proofs of correct shuffling and decryption.

Degree of Verifiability: The shuffle-based approach is *verifiable*.

Further Properties and further Criteria

Similar to the homomorphic approach, *eligibility* and *uniqueness* cannot be evaluated without further details on the identification and authentication mechanism. Shuffles require a significant number of computations for shuffle nodes, key trustees as well as public observers (which might also be each individual voter). In a shuffle, each shuffle node has to operate on the entire set of ciphertexts and provide proofs for the correct operation. To that end, re-encryption shuffles are among the most *costly* approaches to ensure secrecy in electronic voting.

Combination of Phases

In the final part of the analysis, we consider techniques that ensure secrecy and verifiability by integrating previously analyzed SnC techniques thereby addressing special-purpose security models.

Civitas

In this section, we present the Civitas system (Clarkson, Chong, & Myers, 2008). Civitas defends secrecy against adversaries that interact with the voter and observe the voter throughout the voting phase. In this system, tallying authorities are also key trustees.

Description

In the pre-voting phase, voter v authenticates towards a set of registration authorities. Each registration authority $i \in \{1, \dots, n\}$ generates a so-called credential share c_v^i that afterwards is used by the voter to cast her vote. Note, that each credential can be used to cast several votes in order to allow vote updating. Each registration authority encrypts its credential share for voter v with the key trustees' public key pk using a multiplicative homomorphic encryption scheme and publishes the resulting ciphertext $\{c_v^i\}_{pk}^{r_i}$ on the bulletin board next to the voter's identity in the electoral roll. The registration authority provides the voter with c_v^i and a designated-verifier proof (convincing only the designated voter) showing that $\{c_v^i\}_{pk}^{r_i}$ is an encryption of c_v^i. Finally, the voter calculates her credential by multiplying all the credential shares received from the different registration authorities:

$$c_v = \prod_{i \in \{1,\dots,n\}} c_v^i.$$

As such, credentials implement the first part of the secrecy ensuring mechanism of Civitas which happens in the pre-voting phase. Once, the voter obtained her credential c_v she can cast her vote v_v in the voting phase by preparing a tuple of the encrypted credential and the encrypted vote

$$\left(\{c_v\}_{pk}^{r_v}, \{v_v\}_{pk}^{r_v'}\right)$$

together with two ZK proofs showing the well-formedness of the vote and that the voter knows both c_v and v_v in order to avoid replay attacks. The tuple is published on the bulletin board. In case of vote updating the new tuple is published in addition. In case a voter is forced by a coercer to forward her credential, she can replace one credential share, e.g., c_v^1 by a random share c_r, and generate a fake credential

$$c_v^f = c_r \cdot \prod_{i \in \{2,\ldots,n\}} c_v^i.$$

The coercer can then also publish a vote using c_v^f on the bulletin board. In the post-voting phase, the threshold set of key trustees conduct the second part of the secrecy ensuring mechanism: The encrypted credentials published by the registration authorities in the pre-voting phase on the bulletin board and the encrypted credentials associated to cast votes from the bulletin board satisfy the following property due to the underlying encryption scheme:

$$\left\{ \prod_{i \in \{1,\ldots,n\}} c_v^i \right\}_{pk}^{r} = \prod_{i \in \{1,\ldots,n\}} \{c_v^i\}_{pk}^{r_i}$$

In the following, the cast credential and the encrypted credential composed from the encrypted credential shares are shuffled. Once these encrypted credentials are shuffled, the authority needs to identify which cast votes are associated to valid credentials and which to fake credentials without violating secrecy. Therefore, the distributed threshold set of key trustees conducts a verifiable plaintext equivalence test due to (Jakobsson & Juels, 2000) for each cast (shuffled) credential against all valid (shuffled) credentials. If a cast credential is not valid, the associated vote is discarded. Finally, votes associated to valid credentials are distributively and verifiably decrypted as presented in the section on distributed ElGamal decryption.

Security Model

Adversary Model: In order to defend *secrecy* against coercers observing the voter during the vote casting process, Civitas relies on the assumption that voters are not under adversarial control in the pre-voting phase (C7, C8: pre-voting). In order to cast her intended selection throughout the voting phase, it must be assumed that there is a moment in the voting phase in which the adversary cannot control the voter. Hence, the adversary cannot notice the usage of the channel between the voter and her voting environment throughout the complete voting phase (C5: not entire voting phase). Civitas relies on the fact that there is at least one registration authority that is fully trusted (C6) and the adversary cannot observe the communication channel between the voter and the trusted registration authority (C5). Otherwise, if the adversary after the pre-voting phase obtains all key material from the voter, the adversary could decrypt the communication between the voter and that authority and therefore obtain the voter's real credential. Furthermore it must be assumed that a threshold set of key trustees is trusted (C6) in order to not illegitimately decrypt votes, associated credentials, and/or valid credentials posted by the registration authorities together with voter's names in the electoral role. Finally, the system assumes that the adversary does not control the voter's voting environment (C9); otherwise the environment could forward the voter's selection or even her real credential thereby violating secrecy. In (Neumann & Volkamer, 2012), Civitas has been practically improved with respect to secrecy. In the improved version, assumptions (C7, C8: pre-voting) and (C5) are instantiated by a supervised registration authority.

Civitas ensures *fairness* by a threshold encryption scheme analogous to the *homomorphic cryptosystem* and *shuffle* techniques; hence the threshold set of key trustees must be trusted (C6). Furthermore, a malicious voting environment (C9) might release the voter's real credential together with the vote that was cast with that credential.

Civitas ensures *cast-as-intended integrity* under the assumption that the voting environment is trusted (C9) with respect to storing the voter's real credential. Hence, a manipulated voting

environment could assign an invalid credential to the voter's intended vote. Civitas allows voters to verify that their cast vote is stored unalteredly on the bulletin board, such that *stored-as-cast integrity* does not rely on adversarial assumptions. *Tallied-as-stored integrity* is given without adversarial restrictions due to corresponding ZK proofs.

Degree of Verifiability: As shown above, Civitas provides *stored-as-cast* and *tallied-as-stored* verifiability; hence, two out of three properties are verifiable.

Further Properties and further Criteria

Under the assumption that at least half of the registration authorities are trustworthy, ineligible voters cannot obtain credentials (Shirazi, Neumann, Ciolacu, & Volkamer, 2011). Alternatively, voters could forward their real credentials thereby allowing the adversary to vote on behalf of them. Essentially, this is prevented by the use of real and fake credentials that cannot be distinguished by the adversary by the assumptions made for secrecy. Hence, Civitas tends to ensure *eligibility* and *uniqueness* in particular strong interpretations. Civitas ensures secrecy under specific adversary thereby posing an immense computational workload on the key trustees. The computationally most costly part boils down to the removal of unauthorized votes. There have been considerable amount of work on improving complexity of the tallying phase in (Weber, Araujo, & Buchmann, 2007; Spycher et al., 2011). Furthermore, voters are in charge of many computations, starting from distributed credential share acquisition and credential composition along with ZK proof verifications, up to the ballot preparation and the corresponding ZK proofs showing the proper behavior of the voter.

Pretty Good Democracy

In this section, we discuss Pretty Good Democracy (PGD) as presented in (Ryan & Teague, 2009), a combination of code voting and shuffles. PGD is based on the code voting technique, but improves secrecy with respect to voters intending to prove how they voted. Note, there exists another proposal - Pretty Understandable Democracy (PUD) proposed by Budurushi et al. (2013) - which also combines code voting and shuffles Due to the fact that PUD is not yet widely established in the scientific literature, we refrain from its analysis and decided to focus on the analysis of PGD.

Description

In the pre-voting phase, the voting authority generates $k \cdot n \cdot (m+1)$ random codes, where k is the parameter of additionally generated codebooks used throughout the auditing process, n is the numbers of voters and m is the number of candidates, and encrypts each code with the key trustees' common public key. These encrypted codes contain voting codes and acknowledge codes and are posted on the bulletin board. In the following, the encrypted codes are verifiable shuffled by the key trustees and a table P of encrypted codes is constructed. Each row of P corresponds to a generated codebook and is given by the codebook's ID, encrypted codes for m candidates and one acknowledge code for that codebook.

$$i, \{c_{i,1}\}_{pk}, \dots, \{c_{i,m}\}_{pk}, \{c_{i,ack}\}_{pk}$$

The registration authority in collaboration with the key trustees distributively decrypts the codebooks and forwards codebooks in sealed envelopes to the returning officer. After auditing random codebooks, the returning officer randomly assigns a codebook to each eligible voter. Thereafter, the ordering of encrypted codes on each codebook in table P is permuted, resulting in the table Q. As opposed to pure shuffling, the permutation of the shuffle must be reconstructed in the post-voting phase to tally the result. Therefore, each key trustee permutes and re-encrypts the codes in the codebook, and homomorphically adds his permutation value to the previous permutation values in that codebook. Hence, the resulting permutation ϕ of encrypted codes on codebook i is stored in the codebook without anybody knowing this permutation. Therefore, after permuting the codebooks in P, rows in the resulting table Q have the following form:

$$i, \{c_{i,\phi(1)}\}_{pk}, \dots, \{c_{i,\phi(m)}\}_{pk}, \{c_{i,ack}\}_{pk}, \{\phi_i\}_{pk}$$

In the voting, phase, the voter authenticates towards the voting server. She afterwards makes her selection and sends her codebook's ID with her selected voting code to the voting server. The voting server encrypts her voting code, generates a ZK proof of knowledge of the code and posts the encrypted voting code together with the ZK proof in the corresponding row on the bulletin board. The key trustees run plaintext equivalence tests between the cast voting code and the codes on the corresponding codebook in table Q. If a match on is found, this code is marked in the codebook on the bulletin board. Finally, the corresponding acknowledge code is distributively decrypted by the key trustees and returned to the voter. In the post-voting phase, for each row the index of the marked code is encrypted and homomorphically added to the encrypted permutation value $\{\phi_i\}_{pk}$. The resulting encryptions encode the selected candidate index. Hence, these encrypted are verifiably shuffled and decrypted by the key trustees.

Security Model

Adversary Model: With respect to the codebook channel and the channel between a voter and the bulletin board, no assumptions must be made regarding *secrecy*. Even a voter forwarding her acknowledge code cannot convince the adversary about her selection due to the fact that there is only one acknowledge code for her codebook. Hence, even publishing the acknowledge code does not violate secrecy. Even an adversary controlling the voting environment cannot break secrecy. However, if the voting server (C6) reveals the voter's identity together with her cast voting code and the registration authority (C6) reveals the candidate assigned to that voting code, secrecy is violated.

The threshold set of key trustees must also be trusted (C6) with respect to *fairness* in order not to decrypt and interpret voting codes throughout the voting phase.

Cast-as-intended integrity relies on the fact that the registration authority (C6) and the voting environment (C9) do not collaborate. If they do so, the voting environment can contact the registration authority to cast a different voting code of that specific codebook rather the voter's voting code. *Stored-as-cast integrity* is not given due to the following attack: If a malicious voting sever (C6) receives a voter's voting code, it might contact the malicious registration authority (C6) to obtain valid voting codes of that specific code sheet. The voting sever would then proceed with a different voting code than the one sent by the voter. *Tallied-as-stored* verifiability is ensured due to the auditing process of the candidate permutation and the verifiable shuffling and verifiable decryption.

Ryan (2011) proposes an improvement towards trust in the registration authority. In that work, the author presents a distributed codebook generation and printing approach, which allows distributed registration authorities to print and distribute individual digits of valid codes to voters such that none of these authorities knows a complete voting code. Thereby, the adversary must only be assumed not to control all registration authorities simultaneously (C6).

Degree of Verifiability: The above integrity analysis shows that only *tallied-as-stored* is verifiable; hence, one out of three sub-properties is verifiable.

Further Properties and further Criteria

In order to allow ineligible voters to cast votes, they must obtain valid codebooks and authenticate towards the voting server. Hence, if the registration authority and the voting server collaborate, *eligibility* and *uniqueness* can be violated. Furthermore, a threshold set key trustees might illegitimately decrypt codebooks and forward them to the voting server that posts a vote for that codebook. The amount of ZK proofs generated throughout the pre-voting and post-voting phase is high. Furthermore, throughout the voting phase, a significant amount of communication is needed among the key trustees in order to run distributed plaintext equivalence tests and distributively decrypt the proper acknowledge codes for all voters. To that end, the computational *cost* of PDG is relatively high.

Conclusion and Future Directions

Due to the wide range of contrary security requirements, electronic voting, and in particular, remote electronic voting goes hand in hand with security and cryptographic techniques. Constantly, new security notions and techniques are proposed to face the dangers of particularly crucial adversarial assumptions. To date, there is an overwhelming amount of works that focus on security and cryptographic techniques in electronic voting systems. However, throughout the literature, from a real-world perspective (e.g., political decision-makers) two disappointing trends can be seen: First, security notions of SnC techniques are often tailored towards the specific techniques, which makes their comparison hard or even impossible. Second, techniques are not considered thoroughly in their real-world environment and decisive factors for their real-world application are not taken into account. Consequently, this leads to a crucial gap between theoretical achievements and practical applications.

To bridge this gap, in this chapter we specify secrecy, fairness, integrity, and verifiability in a restrictive way and present a modular security model that allows evaluating these properties. Based on the presented methodology, we analyze a number of SnC techniques with respect to further properties that go beyond pure security considerations, namely cost. This criterion turns out to be indispensable for the real-world establishment of electronic voting systems. While the focus of this work is on remote electronic voting, most of the techniques discussed here can also find their application in polling station electronic voting.

The insights gained from this work are interesting from a practical point of view. The rigorous interpretation of secrecy, fairness, integrity, and verifiability in electronic voting systems first allows practitioners to determine techniques adequate for their needs and second shows that techniques and systems often claimed to provide verifiability do not fulfill our rigorous interpretation. It turns out that techniques classified into the pre-voting and voting phases face difficulties in detecting illegitimate vote removal. This stems from that fact that manipulated voting environments might generate identical encodings of identical selections such that voters cannot detect the absence of their individual vote. One way of overcoming this drawback might be to involve the voter in the labeling of her vote, e.g., by choosing the randomness used to encrypt her vote. In this case, however it must be emphasized that significant compromises with respect to secrecy must be made because voters might forward their label also to the adversary and consequently prove their vote. Techniques classified into the post-voting phase apparently do not suffer that drawback, as the voter's identity can be associated to her vote until the tallying process begins. Though, in this case, forced-abstention attacks can be mounted.

As final advice, we encourage decision-makers and technical staff involved in the implementation of electronic voting to conduct a threat and risk analysis with respect to their own electoral circumstances in order to identify the proper SnC technique to be implemented. Code voting and its improved derivation Pretty Good Democracy prove to guarantee secrecy and integrity even over manipulated voting environments coming at the cost of a secure channel implementation. Blind signatures have been early implemented for electronic voting systems and prove to be adequate for low-resource devices (e.g. smartcards) while simultaneously offering accessibility towards the general public. Both code voting and blind signatures must however improve towards stored-as-cast verifiability. Homomorphic cryptosystems and shuffles provide verifiability and ensure secrecy under reasonable assumptions. It must however be noted that these techniques come at significant computational and administrative costs and might therefore not be appropriate for all electoral circumstances. Civitas ensures secrecy against adversaries actively influencing the voter throughout the voting phase and is therefore a special-purpose system. Verifiability cannot be ensured by the system while at the same time both administrative and computational effort is overwhelming. In Table 1 we summarize the results of our evaluation.

In the future, SnC techniques must be completely integrated in a systemic perspective, in particular including identification and authentication mechanisms. Only by integrating these mechanisms into SnC techniques, precise security models can be assessed for eligibility and

uniqueness, and for further properties derived from the law such as *anonymity* and *accountability.* Furthermore, only the integration of SnC techniques into systems allows estimating overall costs.

Furthermore, the principle of public nature goes beyond pure verifiability but rather requires that all essential steps of the voting process must be *understandable* to the voter.

Technique	Secrecy	Fairness	Integrity / Verifiability
Code Voting	The adversary cannot read the channel between the registration authority and the voter. The adversary cannot control the registration authority. The adversary cannot obtain information from the voter	The adversary cannot control the registration authority. The adversary cannot obtain information from the voter.	Cast-as-intended: verifiable Stored-as-cast: The adversary cannot control the registration authority and the tallying authority simultaneously. Tallied-as-stored: verifiable
Online Randomized Authentication Token	The adversary cannot determine the origin of messages on the channel between voters and the tallying authority. The adversary cannot control the registration and the tallying authority simultaneously. The adversary cannot obtain information from the voter before the post-voting phase. The adversary cannot control the voting environment.	The adversary cannot read the channel between the voter and the tallying authority. The adversary cannot control the tallying authority. The adversary cannot control the voting environment.	Cast-as-intended: verifiable Stored-as-cast: The adversary cannot control the voting environment and the tallying authority simultaneously. Tallied-as-stored: verifiable
Blind Signatures	The adversary cannot determine the sender of messages towards the tallying authority. The adversary cannot control the voting environment.	The adversary cannot read the channel between the voter and the tallying authority. The adversary cannot control the voting environment. The adversary cannot control the tallying authority.	Cast-as-intended: verifiable Stored-as-cast: The adversary the voting environment and the bulletin board simultaneously. Tallied-as-stored: verifiable
Homomorphic Cryptosystems	The adversary cannot control a threshold set of key trustees. The adversary cannot control the voting environment.	The adversary cannot control a threshold set of key trustees. The adversary cannot control the voting environment.	Cast-as-intended: verifiable Stored-as-cast: verifiable Tallied-as-stored: verifiable
Shuffles	The adversary cannot control all shuffle nodes. The adversary cannot control a threshold set of key trustees. The adversary cannot control the voting environment.	The adversary cannot control a threshold set of key trustees. The adversary cannot control the voting environment.	Cast-as-intended: verifiable Stored-as-cast: verifiable Tallied-as-stored: verifiable.

Technique	Secrecy	Fairness	Integrity / Verifiability
Civitas	The adversary cannot send/obtain information to/from the voter in the pre-voting phase. The adversary cannot notice the usage of the channel between the voter and her voting environment throughout the complete voting phase. The adversary cannot control all registration authorities. The adversary cannot control a threshold set of key trustees. The adversary cannot control the voting environment.	The adversary cannot control a threshold set of key trustees. The adversary cannot control the voting environment.	Cast-as-intended: The adversary cannot control the voting environment. Stored-as-cast: verifiable Tallied-as-stored: verifiable
Pretty Good Democracy	The adversary cannot control the registration authority and the voting server simultaneously. The adversary cannot control a threshold of key trustees.	The adversary cannot control a threshold set of key trustees.	Cast-as-intended: The adversary cannot control the registration authority and voting environment simultaneously. Stored-as-cast: The adversary cannot control the registration authority and voting server simultaneously. Tallied-as-stored: verifiable

Table 1. Overview on Security and Cryptographic Techniques and their Relation to Secrecy, Fairness, Integrity, and Verifiability. Note that the entries in the table blocks are conjunctions, i.e., all assumptions in one block must be satisfied.

Acknowledgements

This work has been developed within the projects ’ModIWa2’ - Juristisch-informatische Modellierung von Internetwahlen and 'VerkonWa' - Verfassungskonforme Umsetzung von elektronischen Wahlen, which are funded by the Deutsche Forschungsgemeinschaft (DFG, German Science Foundation).

References

Adida, B. (2006). *Advances in Cryptographic Voting Systems*. Cambridge, MA, USA: Massachusetts Institute of Technology

Adida, B. (2008). Helios: Web-based Open-Audit Voting. In P.C. van Oorschot (Ed.) Proceedings of the 17th conference on Security symposium (pp. 335-348). Berkeley, CA, USA. USENIX Association

Adida, B., Neff, C.A. (2009). Efficient receipt-free ballot casting resistant to covert channels. In D. Jefferson, J.L. Hall, T. Moran (Eds.), Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE) (pp. 11-11). Berkeley, CA, USA. USENIX Association

Adida, B., Pereira, O., De Marneffe, O. Quisquater, J. (2009). Electing a university president using open-audit voting: Analysis of real-world use of Helios, In D. Jefferson and J.L. Hall and T. Moran (Eds.), In Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE) (pp. 10-10). Berkeley, CA, USA. USENIX Association

Amenaza Technologies Limited, Ingoldsby T. R. (2005). Attack Tree-based Threat Risk Analysis.

Benaloh, J. (2006). Simple Verifiable Elections. In USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop (pp. 5-5). Berkeley, CA, USA

Blum, M., Feldman, P., Micali, S. (1988). Non-Interactive Zero-Knowledge and Its Applications. In Proceedings of the 29th annual ACM Symposium on Theory of Computing (pp. 103-112). ACM Press, New York

Budursuhi, J., Neumann, S., Volkamer, M. (2012). Smart Cards in Electronic Voting - Lessons learned from applications in legally binding elections and approaches proposed in scientific papers. In Proceedings of the 5th Conference on Electronic Voting 2012 (pp. 258-271), LNI GI Series, Bonn.

Budurushi, J., Neumann, S., Olembo, M., Volkamer, M. (2013). Pretty Understandable Democracy. In Proceedings of Eighth International Conference on Availability, Reliability, and Security (pp. 198-207). Washington, DC, USA. IEEE Computer Society

Carlos, M., Martina, J., Price, G., Custodio, R. (2013). An updated threat model for security ceremonies. In Proceedings of the 28th Annual ACM Symposium on Applied Computing (pp. 1836–1843). New York, NY, USA. ACM

Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S. (2012). Malleable Proof Systems and Applications. In D. Pointcheval and T. Johansson (Eds.) Advances in Cryptologoy - 2012: Vol. 4886 Lecture Notes in Computer Science (pp. 281-300). Cambridge, UK, Springer

Chaum, D. (1981). Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM, 24*(2), (pp. 84-90). NY, USA.

Chaum, D. (2001). SureVote: Technical Overview. Proceedings of the Workshop on Trustworthy Elections (WOTE '01)

Chaum, D., Pedersen, Pedersen, T.P. (1992). Wallet Databases with Observers. In E.F. Brickell (Ed.), Advances in Cryptology – CRYPTO 1992: Vol. 740 Lecture Notes in Computer Science (pp. 89-105). London, UK, Springer

Chaum, D., van Heyst, E. (1991). Group signatures. In Advances in Cryptology - Eurocrypt 1991: Vol. 547 Lecture Notes in Computer Science (pp. 257-265). Cambridge, UK, Springer

Clarkson, M.R, Chong, S., Myers, A.C. (2008). Civitas: Toward a Secure Voting System. In IEEE Symposium on Security and Privacy (pp. 354-368). Oakland, CA. IEEE Computer Society

Clarkson, M.R., Myers, A.C. (2005): *Coercion-resistant remote voting using decryption mixes*. In Workshop on Frontiers in Electronic Elections

Cramer, R., Gennaro, R., Schoenmakers, B. (1997). A Secure and Optimally Efficient Multi-Authority Election Scheme. In Advances in Cryptology - Eurocrypt 1997: Vol. 1233 Lecture Notes in Computer Science (pp. 103-118). Konstanz, Germany. Springer

Dolev, D., Yao, A. (1983). On the security of public key protocols. Technical Report. Stanford, CA, USA.

El Gamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley, David Chaum (Eds.): Advances in Cryptology – CRYPTO 1984: Vol. 196 Lecture Notes in Computer Science (pp. 10-18). Santa Barbara, California, USA. Springer

Feldman, P. (1987). A practical scheme for non-interactive verifiable secret sharing. In Proceedings of the 28th Annual Symposium on Foundations of Computer Science (pp. 427-438). Washington, DC, USA. IEEE Computer Society

Fiat, A., Shamir, A. (1986). How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Advances in Cryptology - CRYPTO 1986: Vol. 263 Lecture Notes in Computer Science (pp. 186-194), Santa Barbara, CA, USA. Springer

Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T. (1999). Secure distributed key generation for discrete-log based cryptosystems. Journal of Cryptology, 20(1), (pp. 51-83)

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st annual ACM symposium on Theory of computing (pp. 169-178). ACM, NY, USA

Gjøsteen, K. (2010). Analysis of an internet voting protocol. In Cryptology ePrint Archive, Report 2010/380.

Goldreich, O., Kahan, A. (1995). How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. Journal of Cryptology, 9, (pp. 167-190)

Heather, J., Ryan, P.Y.A, Teague, V. (2010). Pretty good democracy for more expressive voting schemes. In D. Gritzalis, B. Preneel, M. Theoharidou (Eds.): Proceedings of European Symposium on Research in Computer Security: Vol. 6345 Lecture Notes of Computer Science (pp. 405-423). Athens, Greece. Springer

Helbach, J. (2008). Code Voting - Ein Verfahren für Aktiengesellschaften? In: Informatik 2008, Vol. 1, (pp. 417–422)

Helbach, J. (2009). Code Voting mit prüfbaren Code Sheets. In S. Fischer, E. Maehle and R. Reischuk (Eds.): GI Jahrestagung 2009. (pp. 1856-1862)

Jakobsson, M., Juels, A. (2000). Mix and match: Secure function evaluation via ciphertexts. In T. Okamoto (Ed.): Advances in Cryptology - ASIACRYPT 2000: Vol. 1976 Lecture Notes of Computer Science (pp. 162-177). London, UK. Springer

Jakobsson, M., Juels, A., Rivest, R.L. (2002). Making mix nets robust for electronic voting by randomized partial checking. In Proceedings of the 11th USENIX Security Symposium (pp. 339-353), Berkeley, CA, USA. USENIX Association

Joaquim, R., Ribeiro, C., Ferreira, P. (2009). VeryVote: A Voter Verifiable Code Voting System. In Ryan, P.Y.A. Ryan, B. Schoenmakers (Eds.): VOTE-ID 2009: Vol. 5767 Lecture Notes in Computer Science (pp. 106-121). Springer

Karayumak, F., Kauer, M., Olembo, M. M., Volk, T. & Volkamer, M. (2011). User study of the improved Helios voting system interfaces. *STAST* (pp. 37-44). IEEE Computer Society

Klink, A. (2006). Cryptographic Voting Protocols: A Prototype Design and Implementation for University Elections at TU Darmstadt. Diploma Thesis. Darmstadt, Germany

Kremer, S., Ryan, M., Smyth, B. (2010). Election verifiability in electronic voting protocols. In D. Gritzalis, B. Preneel, M. Theoharidou (Eds.): Proceedings of European Symposium on Research in Computer Security: Vol. 6345 Lecture Notes of Computer Science (pp. 389-404). Athens, Greece. Springer

Küsters, R., Truderung, T., Vogt, A. (2012). Clash Attacks on the Verifiability of E-Voting Systems. *IEEE Symposium on Security and Privacy* (pp. 395-409), : IEEE Computer Society

Lambrinoudakis, C., Gritzalis, D., Tsoumas, V., Karyda, M. & Ikonomopoulos, S. (2003). Secure Electronic Voting: the Current Landscape. In D. Gritzalis (ed.), *Secure Electronic Voting*, Vol. 7 (pp. 101-122). Springer

Langer, L. (2010). Privacy and verifiability in electronic voting. Ph.D. Thesis. Darmstadt, Germany

Liu, J.K., Wei, V.K., Wong, D.S (2004). Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups. In H. Wang, J. Pieprzyk, V. Varadharajan (Eds.): Australasian Conference on Information Security and Privacy, ACISP 2004: Volume 3108 Lecture Notes in Computer Science (pp. 325-335), Springer

Lundin, D. (2010). Component based electronic voting systems. In D. Chaum, M. Jakobsson, R. L. Rivest, P.A. Ryan, J. Benaloh (Eds.) Towards Trustworthy Elections: Vol. 6000 Lecture Notes in Computer Science (pp. 260-273). Springer

MacNamara, K., Iedemska, I. (2012). A Survey of Electronic Voting Schemes. Student Project at University of California.

Mercuri, R. (2002) A Better Ballot Box? IEEE Spectrum, Vol. 39, 2002. IEEE Computer Society

Mitrou, L., Gritzalis, D., & Katsikas, S. (2002). Revisiting legal and regulatory requirements for secure e-voting. *Paper presented at the 16th IFIP International Information Security Conference.* Cairo, Egypt. Kluwer Academics Publisher

Moran, T., Naor, M. (2007). Split-Ballot Voting: Everlasting Privacy with Distributed Trust. In Proceedings of the 14th ACM conference on Computer and communications security. (pp. 246-255), New York, NY, USA. ACM

Mursi, M.F.M, Assassa, G.M.R, Abdelhafez, A., Samra, K.M.A. (2013). On the Development of Electronic Voting. International Journal of Computer Applications, 61(16), 1-11

Naor, M. (1991). Bit Commitment Using Pseudo-Randomness. Journal of Cryptology, 4, 151-158

Naor, M., Pinkas, B. (1999). Oblivious Transfer and Polynomial Evaluation. In Proceedings of the 31st Annual ACM Symposium on Theory of Computing (pp. 245 - 254). ACM Press, New York

Neff, A. (2004): Practical high certainty intent verification for encrypted votes.

Neumann, S., Volkamer, M. (2012). Civitas and the Real World: Problems and Solutions from a Practical Point of View. In Proceedings of Seventh International Conference on Availability, Reliability, and Security (pp. 180-185). Washington, DC, USA. IEEE Computer Society

OASIS Standard (2007): Election Markup Language (EML) Version 5.0 Process and Data Requirements

OSCE/ODIHR (2012). Norway: Internet Voting Pilot Project / Local Government Election / 12 September 2011.

Olembo, M., Schmidt, P., Volkamer, M. (2011). Introducing Verifiability in the POLYAS Remote Electronic Voting System. In Proceedings of Sixth International Conference on Availability, Reliability, and Security (pp. 127-134). Washington, DC, USA. IEEE Computer Society

Olembo, M., Kahlert, A., Neumann, S., Volkamer, M. (2012). Partial Verifiability in POLYAS for the GI Elections. In Proceedings of the 5th Conference on Electronic Voting 2012 (pp. 95-109), LNI GI Series, Bonn.

Organization for the Advancement of Structured Information Standards (2007). Election Markup Language (EML) v5.0

Paillier, P. (1999). Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Advances in Cryptologoy - Eurocrypt 1999: Vol. 1592 Lecture Notes in Computer Science (pp. 223-238). Cambridge, UK, Springer

Pedersen, T.P. (1991). A Threshold Cryptosystem without a Trusted Party. In Advances in Cryptologoy - Eurocrypt 1991: Vol. 547 Lecture Notes in Computer Science (pp. 522-526). Cambridge, UK, Springer

Rivest, R., Shamir, A., Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Journal ACM of Communication, 21, 120-126

Rivest, R., Shamir, A., Tauman, Y. (2001). How to leak a secret. In C. Boyd (Ed.): Advances in Cryptology - ASIACRYPT 2001: Vol. 2248 Lecture Notes of Computer Science (pp. 552-565). Springer

Rjašková, Z. (2002). Electronic Voting Schemes. Diploma Thesis. Bratislava, Slovakia

Rössler, T. (2004). e-Voting: A Survey and Introduction. Technical Report

Ryan, P.Y.A., Teague, V. (2009). Pretty Good Democracy. In B. Christianson, J.A. Malcolm, V. Matyas, and M. Roe (Eds.): Proceedings of the 17th International Workshop on Security Protocols. Vol. 7028 Lecture Notes of Computer Science (pp. 111-130). Springer

Ryan, P. Y. A. (2011). Prêt à voter with confirmation codes. In H. Shacham and V. Teague (Eds.), Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE).

Sako, K., Kilian, J. (1995). Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In L. C. Guillou, J.-J. Quisquater (Eds.): Advances in Cryptology - EUROCRYPT 1995. Vol. 921 Lecture Notes of Computer Science (pp. 393-403). Springer

Sandler, D.R., Wallach, D.S. (2008). The case for networked remote voting precincts. In D. L. Dill and T. Kohno (Eds.) Proceedings of the conference on Electronic voting technology (pp. 6-6). Berkeley, CA, USA, USENIX Association

Schnorr, C. (1989). Efficient Identification and Signatures for Smart Cards. In G. Brassard (Ed.), Advances in Cryptology – CRYPTO 1989: Vol. 435 Lecture Notes in Computer Science (239-252). Springer

Shamir, A. (1979). How to Share a Secret. In Communications of the ACM, 22, pp. 612-613

Shirazi, F., Neumann, S., Ciolacu, I., Volkamer, M. (2011). Robust electronic voting: Introducing robustness in Civitas. In Proceedings of International Workshop on Requirements Engineering for Electronic Voting Systems (pp. 47 -55). IEEE Computer Society

Smith, W. (2005). Cryptography Meets Voting. Technical Report

Spycher, O., Koenig, R., Haenni, R., Schläpfer, M. (2011). *A New Approach Towards Coercion-Resistant Remote E-Voting in Linear Time*. In G. Danezis (Ed.) Proceedings of the 15th International Conference on Financial Cryptography and Data Security. Vol. 7035 Lecture Notes of Computer Science (pp. 182-189). Springer

Teague, V., Ramchen, K., Naish, L. (2008). Coercion-resistant tallying for STV voting. In D. L. Dill and T. Kohno (Eds.) Proceedings of the conference on Electronic voting technology (pp. 15-15). Berkeley, CA, USA, USENIX Association

Volkamer, M. (2009). *Evaluation of Electronic Voting - Requirements and Evaluation Procedures to Support Responsible Election Authorities* (Vol. 30). Springer

Weber, S., Araujo, R., Buchmann, J. (2007). On coercion-resistant electronic elections with linear work. In Proceedings of Second International Conference on Availability, Reliability, and Security. Vienna, Austria. IEEE Computer Society

Wikström, D. (2005). A Sender Verifiable Mix-Net and a New Proof of a Shuffle. In B. Roy (Ed.): Advances in Cryptology - ASIACRYPT 2005: Vol. 3788 Lecture Notes of Computer Science (pp. 273-292). Springer