Couponing Scheme Reduces Computational Power Requirements for DSS Signatures

[Published in *Proc. of CardTech/SecurTech*, pp. 99–104, Rockville, MD, USA, 1994, CTST Inc.]

David M'Raïhi and David Naccache

Gemplus Card International
Crypto Team - Information Security Group
1 Place de Navarre, F-95200, Sarcelles CEDEX, France
Email: {100142.3240, 100145.2261}@compuserve.com

Abstract. This paper introduces the new concept of 'use & throw' *DSS authentication coupons*. An authentication coupon is a couple of pre-calculated certificates provided by a trusted centre during a prior on-line communication.
Coupons are 28 bytes long and contain r and a cryptogram of the information to be used during the computation of s. Coupons can be even reduced to exactly 20-bytes if only one new common random number is generated during each loading session.
DSS Certificates embedding information relevant to applications introduce higher security against fraudulent attempt. Certificates are issued by the authority at the time of loading the related set of coupons. The signature protocol enables simple 8-bit microcontrollers to generate fully compatible DSS signatures in less than 200 ms, about 300 ms with the certificate management, by retrieving r and computing s from the coupon with only two 160-bit modular multiplications. Couponing assumes that the signer enters periodically in contact with the authority to refresh the coupon's reserve.

1 The DSA Original Scheme

The U.S. government has proposed a new cryptographic algorithm as a standard to verify electronically the integrity and source of unclassified information. The process is quite simple in its main goal: it works somewhat like a letter sealed in a transparent envelope. Any receiver can verify the integrity of the message and the identity of the sender.

The Digital Signature Standard [1] approval would lead to an efficient information system where electronic documents would be virtually equivalent to paper ones. Authentication based on digital signatures not only assures the receiver that a bank transaction was sent by, say, CityBank from the East Coast to the West Coast, but also the rightness of the message (the message has not been modified by a forger or altered due to a problem on the line).

The Digital Signature Algorithm (DSA), proposed in August 1991 by the US National Institute of Standards and Technology, is a DLP-based cryptosystem which parameters are:

1. A prime modulus p where $2^{L-1} < p < 2^L$ for $512 \leq L \leq 1024$ and $L \bmod 64 = 0$.
2. A prime q such that $2^{159} < q < 2^{160}$ and $p-1$ is a multiple of q.
3. A number $g = h^{(p-1)/q} \bmod p$ for some h.
4. A 160-bit secret-key x and an L-bit public-key y defined by the relation: $y = g^x \bmod p$.

The integers p, q and g are system parameters and can be public and/or common to a group of users. A 160-bit random k, used by the signer, must be kept secret and regenerated for each signature.

In order to sign a message m (hashed value of a primitive file M), the signer computes the signature $\{r, s\}$ by:

$$r = (g^k \bmod p) \bmod q \quad \text{and} \quad s = \frac{m + xr}{k} \bmod q$$

To check $\{r, s\}$, the verifier computes:

$$w = \frac{1}{s} \bmod q, \quad u_1 = mw \bmod q \quad \text{and} \quad u_2 = rw \bmod q$$

And compares if $r = (g^{u_1} y^{u_2} \bmod p) \bmod q$ to accept or reject the signature

Assuming no algorithmic sophistications,[1] the resources necessary for the implementation of the DSA are:

resources	***Signer***	***Verifier***
# of 160 bit mult.	2	2
# of 512 bit mult.	≈237	≈475
modular inverse	yes	yes
transmission	40 bytes	
size of moduli	84 bytes	

And the complete process is briefly summarised in figure 1.

2 Couponing Scheme

A well-known feature of the DSA, inherited from its ancestors **El-Gamal** [3] and **Schnorr** [4], is the possibility to pre-compute r and the inverse of k before the message is known. Then, the effort needed to produce s from m is negligible.

[1] Some of which [2] may spectacularly divide all the 512-bit figures by about 6 but these tools apply exactly in the same manner to our schemes. The important point is the *ratio* between the protocols which remains constant whatever exponentiation strategy is used.

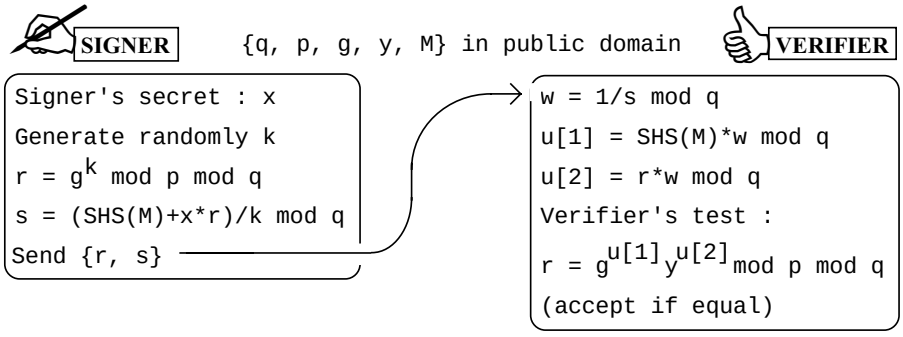

Fig. 1. The Digital Signature Algorithm

This section introduces a coupon-based protocol exploiting this property for helping the signer to generate signatures very quickly. In our model, a trusted authority sends public data packets (*Use & Throw coupons*) to the signer who stores them for future use.

Each coupon is *only* 28-byte long and enables its owner to generate **one** DSA signature (if a coupon is used twice, the signer's x is revealed). Two noteworthy advantages of this method are that the signer has only to possess x and q (the storage of g and p, which represents 1024 bits at minimum, can be avoided) and only a couple of multiplications is needed to transform a coupon to a signature.

The system is based on a retro-calculation of k from an easily compressible inverse and is ideally suited to electronic-purse applications where card-holders interact periodically with a trusted authority for loading money into their purses (refreshing the coupon's reserve):

This scheme was implemented on a 68HC05-based prototype[2] which generates s in less than 150 ms (4 MHz clock) and can contain up to 91 coupons in EEPROM. A *heavy-duty* version (now under development) will be 30% faster and tailored to contain about 400 coupons.

Note that when **Montgomery**'s algorithm [5] is used (let $Q = 2^{-size(q)} \bmod q$), the signer can shortcut his calculations by using the key: $x' = xQ^{-1} \bmod q$ if the authority compensates:

$$r = (\sqrt[q]{g} \bmod p)^k \bmod p \bmod q$$

The coupon-owner will then compute s by two **Montgomery** rounds (instead of four):

1. $z = Montgomery_q(x', r) \equiv x'rQ \equiv xQ^{-1}rQ \equiv xr \bmod q$
2. $s = Montgomery_q(z + m, \mathrm{SHA}(J|x)) \equiv (z + m)\mathrm{SHA}(J|x)Q \bmod q$

The signature is still DSA-compatible and the storage of $4^{size(q)} \bmod q$ (40 bytes normally needed for converting results from **Montgomery's** format to the conventional number system) has been avoided.

[2] ST16623 (no crypto-engine aboard).

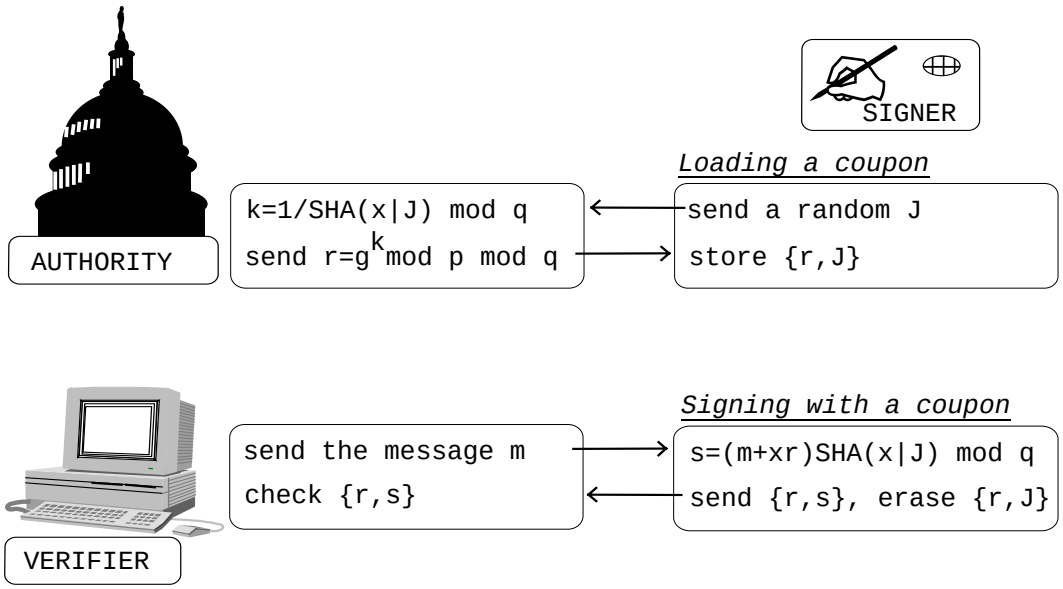

Fig. 2. Couponing Scheme

3 Improvements Regarding Coupon Size

Coupons can be reduced to exactly 20-bytes if only one new common 20-byte long J is generated during each loading session and inverses are diversified by $\mathrm{SHA}(J|x|i)$ where i is the coupon's number.

With such a solution a typical EEPROM map of a coupon-card supporting the Diversified Couponing Scheme will be:

secret x
modulus q
common J
coupon 1
coupon 2
⋮
coupon n

where n, (the number of coupons that can be held in a card) is typically:

$$n = \frac{\text{EEPROM Capacity (in bytes)}}{20} - 3$$

The gain in terms of coupons when compared to the simple couponing scheme is:

$$\begin{aligned}\text{Gain} &= \left(\frac{E-60}{20}\right) - \left(\frac{E-40}{28}\right) = \frac{7E-420-5E+200}{140} = \frac{2E-220}{140} \\ &\approx \left\lfloor \frac{E-110}{70} \right\rfloor \text{ coupons}\end{aligned}$$

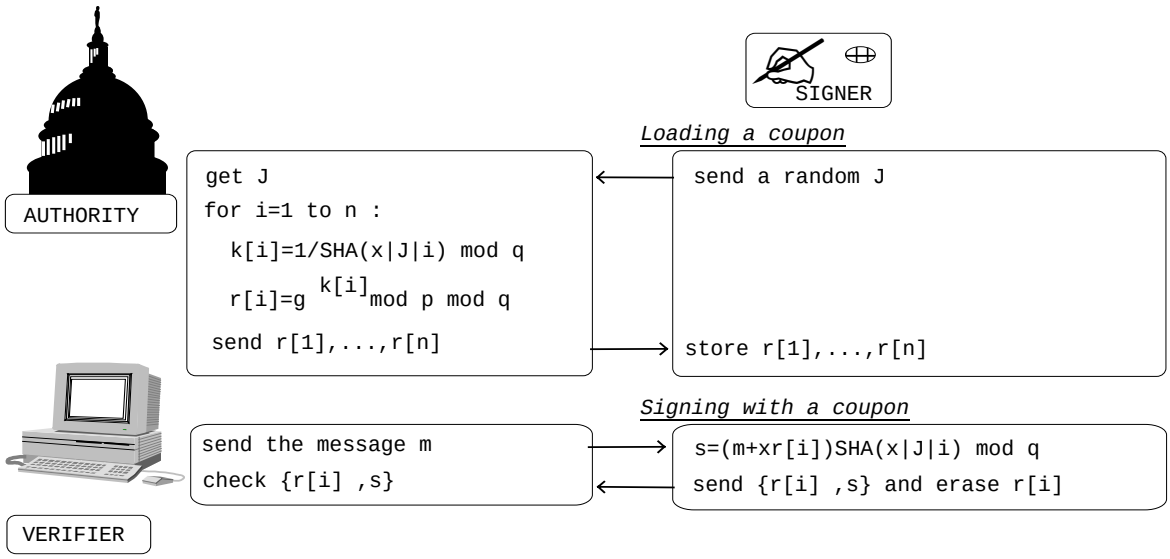

Fig. 3. The diversified Couponing Scheme

thus the Gain is 41 coupons when the EEPROM capacity (denoted E) is 3000 bytes, a second characteristic instance (ST16F48) is $\{E = 8000, \text{Gain} = 112\}$.

The characteristics of a coupon system can be summarised as the combination of:

$$\text{EEPROM Capacity (in bytes)} = 60 + 20n \quad \text{and}$$

$$\text{Transmission (in bytes)} = \text{ISO bytes} + \mathit{size}(m) + 60 \approx 115$$

Assuming a 3000 byte EEPROM (ST16623) and a 115.200 baud interface these relations yield:

$$n = 147 \text{ coupons} \quad \text{and}$$

$$\begin{aligned}\text{Time} &\approx \text{Transmission}(115; 155, 200) + \text{Processing} + \text{Writing in EEPROM} \\ &\approx 200\text{ms}\end{aligned}$$

Asuming a 8000 byte EEPROM and the same transmission time we get:

$$n = \frac{800 - 6}{2} = 397 \text{ coupons}$$

4 Security Improvement

It is suitable to limit the validity of a coupon to a certain date or geographical area. For doing so, the authority associates to a set of coupons a DSA signature (40 bytes) which has to be checked first by the verification terminal before going further on. This increases the size of the coupons to $\lfloor 20 + \frac{40}{n} \rfloor$ bytes and the memory map becomes:

secret x
modulus q
common J
coupon 1
coupon 2
⋮
coupon n
certif $1, ..., n$

This solution introduces an additional transmission overhead since the card has to present (without signing) n coupons along with the signature. The problem is therefore to get the balance right, regarding transmission and memory requirements.

If coupons are certified by groups of c and the card contains n coupons we have:

$$\text{EEPROM Capacity (in bytes)} = 60 + 20n + \frac{40n}{c} \quad \text{and}$$

$$\text{Transmission (in bytes)} = \text{ISO bytes} + size(m) + 8c + 12 + 60 \approx 8c + 115$$

Assuming a 3000 byte EEPROM (ST16623) and a 115.200 baud interface these relations become:

$$147 = n\left(1 + \frac{2}{c}\right) \quad \text{and}$$

$$\begin{aligned}\text{Time} &\approx \text{Transmission}(8c + 115; 155, 200) + \text{Processing} + \text{Writing in EEPROM} \\ &\approx 0.85c + 200\text{ms}\end{aligned}$$

If we assume in advance a global transaction time of 300 ms (SHA is included therein), we get $c \approx 70$ which yields $n = 142$ coupons. An interesting instance of the system for implementation on a smartcard would therefore be a two coupon set version, with each set containing 70 coupons and a certificate for a total of 140 coupons.

The same evaluation on a bigger component with a 8000 byte EEPROM, assuming a transaction time of 300 ms and so $c < 120$, gives the following relations:

$$\begin{cases} c \in [1, 120] \\ 8000 = 60 + 20n + \frac{40n}{c} \end{cases} \iff \begin{cases} c \in [1, 120] \\ n = \frac{397c}{c+2} \Rightarrow n \in [132, 390] \end{cases}$$

Let $c|n$ then a couple of integer solutions to maximize n is $\{c, n\} = \{118, 354\}$.

5 Conclusion

As a conclusion, it is possible to design a coupon-system (capacity: about 140 coupons) which computes fully-compatible DSA signatures in less than 200 ms within only a 3K byte EEPROM. The choice of bigger components such as the

ST16F48 (8K EEPROM) leads to heavy-duty versions, with 279 coupons for a 28-byte coupon implementation, more than 400 coupons for a 20-byte coupon solution, depending on time, memory and security requirements.

Besides, generating a signature from a coupon is much more restrictive than doing the same freely with the basic DSA as coupons has to be presented with a certificate that may limit their validity in time or territory. As coupon-based signatures are fully DSA-compatible but restricted, employing them limits the risk of breaking a card and extracting from it the coupons and the secret for double-spending.

References

1. FIPS PUB XX, February 1, 1993, *Digital Signature Standard*.
2. E. Brickell, D. Gordon and K. McCurley, *Fast exponentiation with precomputation*, technical report no. SAND91-1836C, Sandia National Laboratories, Albuquerque, New-Mexico, October 1991.
3. T. El-Gamal, *A public-key cryptosystem and a signature scheme based on discrete logarithms*, IEEE TIT, vol. IT-31:4, pp 469–472, 1985.
4. C. Schnorr, *Efficient identification and signatures for smart-cards*, Advances in Cryptology: Proceedings of Eurocrypt'89 (G. Brassard ed.), LNCS, Springer-Verlag, Berlin, 435 (1990), pp. 239–252.
5. P. Montgomery, *Modular multiplication without trial division*, Mathematics of Computation, vol. 44(170), pp. 519–521, 1985.