Evaluation of Digital Signature Process

Emil SIMION, Ph. D.

email: esimion@fmi.unibuc.ro

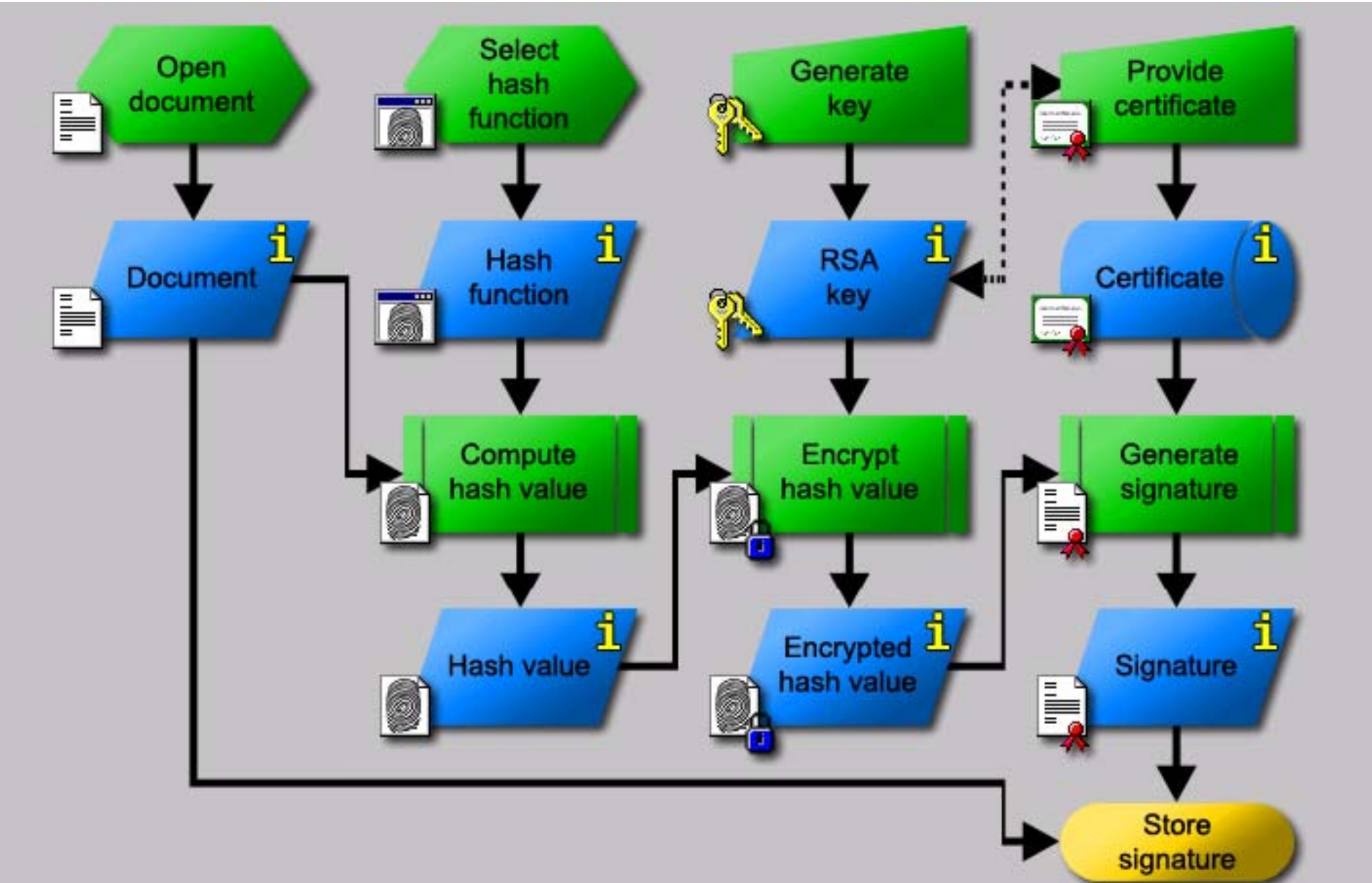
Agenda

- Evaluation of digital signatures schemes:
  - evaluation criteria;
  - security evaluation;
  - security of hash functions;
  - security of RSA, DSA and ECDSA schemes;
  - case study transition from FIPS 186-2 to FIPS 186-3.
- Vulnerabilities in digital signatures schemes:
  - main attacks on RSA;
  - case study: RSA and CRT;
  - case study: padding in PCKS-1, X9.31;
  - ISO/IEC 9796.
- Others features of digital signatures schemes:
  - Cryptographic protocols;
  - Subliminal channels in ElGamal and DSA signatures.

Evaluation of digital signatures scheme

- There are three area to evaluate:
  - Evaluation of the algorithm;
  - Evaluation of the implementation;
  - Evaluation of the device (e.g. token) and the scheme;
  - Evaluation of the system (risk analysis).
- Methodology:
  - Academic evaluation (e.g. during international conferences or R&D programs);
  - Standardization (e.g. by NIST, ISO, NSA).

General scheme of RSA digital signature process



Security evaluation

- Security evaluation is done on:
  - Hash function, which provides a digest to be encrypted using private key;
  - Public key algorithm (RSA, DSA, ECDSA etc.);
  - Public key generation process;
  - Public Key Infrastructure (PKI).

Security of hash functions

Hash functions *H* must be one-way:

- given the value x, it is easy to compute $H(x)$;
- given $H(x)$, it is computationally difficult to compute the input x.

Examples: SHA family standardized in *FIPS 180-2, Secure Hash Standard* (SHS) lacks in security:

- In 2005 Prof. Xiaoyun Wang announced a differential attack on the SHA-1 hash function; with her recent improvements, this attack is expected to find a hash collision (two messages with the same hash value) with an estimated work of 2^{63} operations, rather than the ideal 2^{80} operations that should be required for SHA-1 for any good 160-bit hash function.

New functions: NIST has selected **five SHA-3 finalists**- BLAKE, Grøstl, JH, Keccak, and Skein to advance to the third (and final) round of the SHA-3competition.

A one-year public comment period is planned for these candidates. NIST also plans to host a final **SHA-3 Candidate Conference** in the spring of 2012 to discuss the public feedback on these candidates, and select the SHA-3 winner later in 2012.

Security of RSA, DSA and ECDSA

- RSA security is based on the difficulty of factorizations of large numbers: given *n* (big number) find the prime factors;
- DSA security is based on the difficulty of solving discrete log problem in prime fields: given *g*, *x* and *p* (prime) find y such $x=g^x \bmod p$.
- ECDSA security is based on the difficulty of solving discrete log problem in EC group (over primes or over GF).

Transition from FIPS 186-2 to FIPS 186-3

- FIPS 180-3 Secure Hash Standard (SHS) (Oct. 2008):
  - This Standard specifies five secure hash algorithms - SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 - for computing a condensed representation of electronic data (message).
- FIPS 186-3 Digital Signature Standard (DSS) (June 2009):
  - The FIPS specifies three techniques for the generation and verification of digital signatures that can be used for the protection of the datat: the Digital Signature Algorithm (DSA), the Elliptic Curve Digital Signature Algorithm (ECDSA), and Rivest Shamir Adelman (RSA) algorithm Although all three of these algorithms were approved in FIPS 186–2, FIPS 186–3 increases the key sizes allowed for DSA, provides additional requirements for the use of RSA and ECDSA, and includes requirements for obtaining the assurances necessary for valid digital signatures.

Vulnerabilities in digital signatures schemes

Main attacks on RSA

- **Chosen ciphertext attack:**
  - **Some attacks work against the implementation of RSA**. These are not attacks against the basic algorithm, but against the protocol. It's important to realize that it's not enough to use RSA. This attack and can be avoided if we use a one-way hash function before signing a document.
- **Common Modulus Attack on RSA**
  - A possible RSA implementation gives everyone the same n, but different values for the exponents *e* and *d*. Unfortunately, this doesn't work. The most obvious problem is that if the same message is never encrypted with two different exponents (both having the same modulus), and those two exponents are relative prime (which they generally would be), then the plain text can be recovered without either of the decryption exponents. This attack is feasible if we use a common *n* among a group of users.

Main attacks on RSA – cont 1.

- **Low encryption exponent attack against RSA**
  - RSA encryption and signature verification are faster if we use a low value for e, but that can also be insecure. If you encrypt $e(e+1)/2$ linearly dependent messages with different public keys having the same value of e, there is an attack against the system. If there are fewer than that many messages, or if the messages are unrelated, there is no problem. If the messages are identical, then e messages are enough. The easiest solution is to pad messages with independent random values. Most real-world RSA implementations –PEM and PGP for example –do this. To avoid this kind of attack the messages must be padded with random values before encrypting them; make sure that m is about the same size as n.
- **Low decryption exponent attack against RSA**
  - Another attack, this one by Michael Wiener, will recover d, when d is up to one quarter the size of n and e is less than n. This rarely occurs if e and d are chosen at random, and cannot occur if e has small value. This attack can be avoided if we chose a large value for d.

Main attacks on RSA – cont 2.

- **Attack on encryption and signing with RSA**
  - It makes sense to sign a message before encryption it, but not everyone follows this practice. With RSA, there is an attack against protocols that encrypt before signing.
- **Attack in case of small difference between prime numbers *p* and *q***
  - In the situation that the prime numbers *p* and *q* are close one to each other the following remark allows us to develop an efficient searching algorithm for *p* and *q*:

$$\left(\frac{p+q}{2}\right)^2 - n = \left(\frac{p-q}{2}\right)^2.$$

Side channel (hardware) attack in RSA

- Hardware attacks exploits some hardware parameters such running time, simple power analysis (SPA), differential power analysis (DFA) and fault analysis (FA).
- Examples of hardware attacks are:
  - *Timings attacks*: depending the running time we can predict which of the bits from the key are zero and which of the bits of the key are one;
  - *SPA attack*: depending the consume power of the cryptographic engine and using adequate math we can derive the key bits. This attack is suitable for crypto devices with external power supply such as smart cards, using the consumed power we can recover the code source from the inside of the smart card;
  - *DPA attack*: in the most cases of microprocessors the consumed power depends on the value of the operand (for example erasing a bit requires a less power then setting a bit), measuring different inputs we can deduce the value of the operand;
  - *Fault analysis*: we can induce same faults in the processor computations and using some math we can derive the key bits.

Case study: RSA and CRT

- Timing attacks: Expose private information, such as RSA keys, by measuring the amount of time required to perform private key operations (decryptions).
- Side-channel attacks:
  - Power analysis
  - Electromagnetic radiation analysis
  - Timing attacks
  - Fault injections

RSA quick review

- Multiple prime RSA key generating algorithm

  1. Select k primes: $p_1, p_2, \ldots, p_k$
  2. Let $n=\prod p_i$, $i=1,2,\ldots,k$
  3. Let $\varphi(n)=\prod(p_i-1)$
  4. Choose e, s.t. $\gcd(e,\varphi(n))=1$
  5. Calculate $d=e^{-1}(\text{mod } \varphi(n))$
  6. Public Key = (e,n) and Private key = (d,n)

- Encryption: $c=m^e \bmod n$
  Decryption: $m=c^d \bmod n$

Chinese Remainder Theorem

- $n = n_1 n_2 \ldots n_k$ with gcd $(n_i; n_j) = 1$ *when* $i \neq j$.
- The system of congruencies

  $x \equiv x_1 \pmod{n_1} \equiv \ldots \equiv x_k \pmod{n_k}$

  has a simultaneous solution x to all of the congruencies, and there exists exactly one solution x between 0 and n-1.

Speedup RSA with CRT

- Any message $M<N$ is uniquely represented by the tuple $[M_p;M_q]$, where

  $M_p = M \bmod p$ and $M_q = M \bmod q$.

  $C_p = C \bmod p$ and $C_q = C \bmod q$.

  $D_p = d \bmod (p-1)$ and $D_q = d \bmod (q-1)$

  $R_p = q^{p-1} \bmod N$ and $R_q = p^{q-1} \bmod N$

  $M_p = C_p{}^{d_p} \bmod p$ and $M_q = C_q{}^{d_q} \bmod q$

  $S_p = M_p R_p \bmod N$ and $S_q = M_q R_q \bmod N$

  $M = S_p + S_q$. If $M>=N$ then calc $M=M-N$.

Operations needed for Decryption

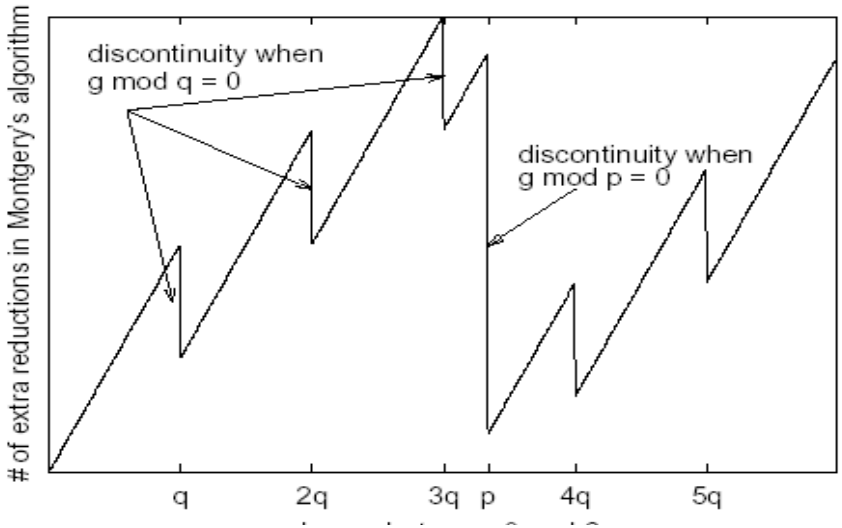
- Computing $c^d \pmod{p}$ and $xy \pmod{p}$ requires:
  - **Multiplication routines**
    - Normal (unequal len)
    - Karatsuba (equal len): faster
  - **Exponentiation**
    - Sliding windows
  - **Modular reduction**
    - Montgomery
    - The key relevant fact is the extra reduction

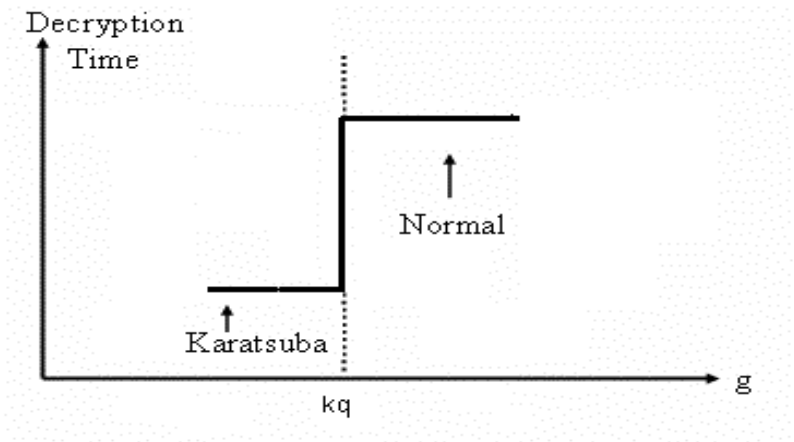
What causes time variance?

- **Montgomery reduction**
  - Given g calc g mod q
  - Probability for and extra reduction is:

  P[extra step] ≈ (g mod q)/2q

- **Choice of multiplication routine**
  - To calc xg mod q, if x is the same length as (g mod q), use Karatsuba. $O(n^{\log_2 3})$
  - Otherwise, use Normal. $O(nm)$





Summary of time variance

	$g<q$	$g>q$
Montgomery effect	Longer	Shorter
Multiplication effect	Shorter	Longer
g is the decryption value. Each is dominant at a different phase.		

Case study: padding in PCKS-1, X9.31

- **An attack has been found on some implementations of RSA** digital signatures using the padding scheme of PCKS-1 when the public key $e = 3$. Details of the attack on PKCS-1 implementations are provided below. A similar attack could also be applied to implementations of digital signatures as specified in American National Standard (ANS) X9.31. Note that this attack is not on the RSA algorithm itself, but on improper implementations of the signature verification process.
- **The testing used by the Cryptographic Algorithm Validation Program (CAVP) has been modified to test for this improper implementation on new RSA signature** applications that undergo CMVP testing. However, many implementations that have already been certified as implementing RSA signatures using PKCS-1 might have been implemented with this problem.
- **NIST has designed a sequence of messages** that can be used by a vendor to test the vulnerability of an implementation to this type of attack.
- **Details**: http://csrc.nist.gov/groups/ST/toolkit/documents/dss/RSAstatement_10-12-06.pdf

Case study: ISO/IEC 9796

- **ISO/IEC 9796-2**: 2002 specifies three digital signature schemes giving message recovery, two of which are deterministic (non-randomized) and one of which is randomized. The security of all three schemes is based on the difficulty of factorizing large numbers. All three schemes can provide either total or partial message recovery.
- In 1999, Coron, Naccache and Stern **discovered an existential signature forgery for two popular RSA signature standards, ISO/IEC 9796-1 and 2.** Following this attack ISO/IEC 9796-1 was withdrawn. ISO/IEC 9796-2 was amended by increasing the message digest to at least 160 bits. Attacking this amended version required at least 2^{61} operations.
- Naccache (*http://eprint.iacr.org/2009/203.pdf*), exhibit algorithmic refinements allowing to attack the amended (currently valid) version of ISO/IEC 9796-2 for all modulus sizes. A practical forgery was computed in only two days using 19 servers on the Amazon EC2 grid for a total cost of aprox. $800. The forgery was implemented for $e = 2$ but attacking odd exponents will not take longer. The forgery was computed for the RSA-2048 challenge modulus, whose factorization is still unknown.

DEMO: RSA attacks

- Simulations are made with CRYPTOOL:
  - Ulrich Kuehn's side-channel attack;
  - Attack on small private exponents.

Others features of digital signatures

- **Cryptographic protocols**:
  - Authentication protocols;
  - Key agreement protocols.
- **Subliminal channels**:
  - Subliminal channels in digital signature crypto systems were found in 1984 by Gustavus Simmons. They are a subgroup of Covert channels and can be used to communicate secretly in a normal looking communication over an insecure channel with help of digital signatures.
  - Simmons describes how the dilemma can be solved through parameter substitution in digital signature algorithms. In signature algorithms like ElGamal and DSA exist parameters which have to be set with random. He shows how one can make usage of these parameter to send a message subliminally. Because the algorithm's signature creation procedure is unchanged, the signature remains verifiable and indistinguishable from a normal signature. Therefore it is hard to detect if the subliminal channel is used.

Thank you for your attention!