Computer Science 418
Digital Signatures

Mike Jacobson

Department of Computer Science
University of Calgary

Week 12

Outline

Digital Signatures: Definition

Data origin authentication is usually achieved by means of a *signature*, i.e. a means by which the recipient of a message can authenticate the identity of the sender.

Definition 1 (Digital signature)

A means for data authentication that should have two properties:

1. Only the sender can produce his signature.
2. *Anyone* should be easily able to verify the validity of the signature.

Digital Signatures: Observations

Observations:

- Properties 1 and 2 provide *non-repudiation:* if there is a dispute over a signature (a receiver claims that the sender signed the message, whereas the signer claims he didn't), anyone can resolve the dispute. For ordinary written signatures, one might need a hand-writing expert.
- Signatures are different from MACs:
  - both sender and receiver can generate a MAC, whereas only the sender can generate a signature.
  - only sender and receiver can verify a MAC, whereas anyone can verify a signature.
- In order to prevent *replay attacks* (replay a signed message later), it may be necessary to include a time stamp or sequence numbers in the signature.

Signature Capable PKCs

Definition 2 (Signature capability)

A PKC is *signature capable* if $\mathcal{M} = \mathcal{C}$ and $E_{K_1}(D_{K_2}(C)) = C$ for all $C \in \mathcal{C}$.

So in a signature capable PKC, decryptions are right and left inverses (*i.e.* honest-to-goodness inverses) of encryptions.

Example 3

RSA has signature capability. ElGamal and Goldwasser-Micali do not.

Signatures Without Secrecy Using PKC

Alice wishes to send a non-secret message M to Bob along with a signature S that authenticates M to Bob.

She sends (A, M, S) where

- A is her identity,
- M is the message,
- $S = D_A(M)$ is the "decryption" of M under her private key.

To verify S, Bob

- checks A and looks up Alice's public key,
- computes the "encryption" $E_A(S)$ of S under Alice's public key,
- accepts the signature if and only if $M = E_A(S)$

Note that $E_A(S) = E_A(D_A(M)) = M$ if everything was done correctly.

Properties

Anyone can verify a signature since anyone can encrypt under Alice's public key.

In order to forge a signature of a particular message M, Eve would have to be able to do decryption under Alice's public key.

Signatures With Secrecy Using PKC

Alice wishes to send an authenticated secret message M to Bob.

She sends $(A, E_B(S, M))$ where A and S are as before and E_B denotes encryption under Bob's public key.

To verify S, Bob decrypts $E_B(S, M)$ and then verifies S as before.

Security of Signatures

Definition 4 (Existential forgery)

A signature scheme is susceptible to *existential forgery* if an adversary can forge a valid signature of another entity for at least one message.

Goals of the attacker:

- total break — recover the private key
- universal forgery — can generate a signature for any message
- selective forgery — can generate a signature for some message of choice
- existential forgery — can generate a signature for at least one message

Existential Forgery on PKC-Generated Signatures

Consider generating a signature S to a message M using a signature-capable PKC as described above.

Eve can create a forged signature from Alice as follows:

1. Selects random $S \in \mathcal{M}$,
2. Computes $M = E_A(S)$,
3. Sends (A, M, S) to Bob.

Bob computes $E_A(S)$ which is M and thus accepts the "signature" S to "message" M.

Usually foiled by language redundancy, but may be a problem is M is random (eg. a cryptographic key).

Preventing Existential Forgery

Solution:

- Alice sends $(A, M, S = D_A(H(M)))$ where H is a public pre-image resistant hash function on $\mathcal{M}$.
- Bob computes $E_A(S)$ and $H(M)$, and accepts the signature if and only if they match.

Foils the attack:

- if Eve generates random S, then she would have to find X such that $H(X) = M = E_A(S)$ (*i.e.* a pre-image under H), and send (A, X, A) to Bob.
- Bob then computes $D_A(H(X))$ and compares with L.
- Not computationally feasible if H is pre-image resistant.

Existential Forgery if H is not Collision Resistant

Suppose Alice uses a pre-image resistant hash function as described above to sign her messages.

If H is not collision resistant, Eve can forge a signature as follows:

1. Find $M, M' \in \mathcal{M}$ with $M \neq M'$ and $H(M) = H(M')$ (a collision)
2. If S is the signature to M, then S is also the signature to M', as $E_A(S) = H(M) = H(M')$

Note that if Eve intercepts (A, M, S), then she could also find a weak collision M' with $H(M) = H(M')$.

Summary on Signatures via PKC

Use a secure signature capable PKC and a cryptographic (*i.e.* collision resistant) hash function H (security depends on both).

Signing $H(M)$ instead of M also results in faster signature generation if M is long.

H should be a fixed part of the signature protocol, so Eve cannot just substitute H with a cryptographically weak hash function.

GMR-Security

In practice, signature schemes must be resistant to active attacks. We need the equivalent of IND-CCA2 for signatures.

Definition 5 (GMR-security)

A signature scheme is said to be *GMR-secure* if it is existentially unforgeable by a computationally bounded adversary who can mount an adaptive chosen-message attack.

In other words, an adversary who can obtain signatures of any messages of her own choosing from the legitimate signer is unable to produce a valid signature of any new message (for which it has not already requested and obtained a signature) in polynomial time.

GMR stands for *Goldwasser-Micali-Rivest*.

GMR-Secure Versions of RSA

Example 6

RSA-PSS (Probabilistic Signature Scheme), a digital signature analogue of OAEP, is GMR-secure in the random oracle model (ROM) assuming that the RSA problem (computing eth roots modulo n) is hard.

Example 7

RSA with *full-domain hash* — use RSA signatures as usual, signing $H(M)$, but select the hash function H such that $0 \leq H(M) < n$ (n is the RSA modulus) for all messages M.

- Called full-domain because the messages signed are taken from the entire range of possible RSA blocks as opposed to a smaller subrange.
- Also GMR-secure under same assumption as above.

Other Signature Schemes

Examples of non-PKC-based signature schemes:

- ElGamal — randomized, security based on DLP
- Digital Signature Algorithm — variation of ElGamal with short signatures
- Feige-Fiat-Shamir — security based on computing square roots modulo pq
- Guillou-Quisquater — security based on the RSA problem of computing e-th roots modulo pq

We'll cover the first two here.

Solving General Linear Congruences

We need to solve a general linear congruence of the form

$$ax \equiv b \pmod{m}$$

for $x \in \mathbb{Z}_m^*$, with $m \in \mathbb{N}$ and $a \in \mathbb{Z}_m^*$.

We already saw how to do this for $b = 1$; that's just finding modular inverses.

To solve $ax \equiv b \pmod{m}$ for x : first solve $az \equiv 1 \pmod{n}$ for z using the Extended Euclidean Algorithm. Then $x \equiv bz \pmod{n}$ as

$$ax \equiv a(bz) \equiv (az)b \equiv 1 \cdot b \equiv b \pmod{n} .$$

The El Gamal Signature Scheme

The El Gamal signature scheme is a variation of the El Gamal PKC (same 1985 paper). Security considerations are the same.

A produces her public and private keys as follows:

1. Selects a large prime p and a primitive root g of p.
2. Randomly selects x such that $0 < x < p - 1$ and computes $y \equiv g^x \pmod{p}$.

Public key: $\{p, g, y\}$
Private key: $\{x\}$

A also fixes a public cryptographic hash function $H : \{0,1\}^* \mapsto \mathbb{Z}_{p-1}$.

Signing and Verifying

A signs a message $M \in \{0,1\}^*$ as follows:

1. Selects a random integer $k \in \mathbb{Z}_{p-1}^*$.
2. Computes $r \equiv g^k \pmod{p}$, $0 \le r < p$.
3. Solves $ks \equiv [H(M \| r) - xr] \pmod{p-1}$ for $s \in \mathbb{Z}_{p-1}^*$
4. A's signature is the pair (r, s).

B verifies A's signature (r, s) as follows:

1. Obtains A's authentic public key $\{p, g, y\}$.
2. Verifies that $1 \le r < p$; if not, reject.
3. Computes $v_1 \equiv y^r r^s \pmod{p}$ and $v_2 \equiv g^{H(M\|r)} \pmod{p}$.
4. Accepts the signature if and only if $v_1 = v_2$.

Proof of Correctness

Proof of correctness.

Note that $ks + rx \equiv H(M, r) \pmod{p-1}$. If the signature (r, s) to message M is valid, then

$$\begin{aligned} v_1 &\equiv y^r r^s \\ &\equiv (g^x)^r (g^k)^s) \\ &\equiv g^{xr+ks} \\ &= g^{H(M\|r)} \\ &\equiv v_2 \pmod{p} . \end{aligned}$$

□

Example

Let $p = 467$, and set $g = 2$ which is a primitive root modulo 467.

- Choose the secret key $x = 127$.
- Using binary exponentiation, one obtains $y \equiv 2^{127} \equiv 132 \pmod{467}$.

So consider an ElGamal user Alice with

- public key $\{467, 2, 132\}$
- private key 127.

Example: signature generation

Suppose Alice wishes to sign the message $M =$ "Hi there".

- She selects $k = 213$; note that $\gcd(213, 466) = 1$.
- Binary exponentiation yields $r \equiv 2^{213} \equiv 29 \pmod{467}$.

Suppose our hash function yields $H(\text{"Hi there"}\|29) = 100$.

- Alice needs to solve

$$123s \equiv 100 - 127 \cdot 29 \equiv 145 \pmod{466} .$$

- First solve $123z \equiv 1 \pmod{466}$ for z using the Extended Euclidean Algorithm, obtaining $z \equiv 431 \pmod{466}$.
- Then $s \equiv 145 \cdot 431 \equiv 51 \pmod{466}$.
- The signature to "Hi there" is $(r, s) = (29, 51)$.

Example: verification

To verify this signature, first note that $r = 29 < 467$. Then compute

$$v_1 \equiv 132^{29} \cdot 29^{51} \equiv 189 \pmod{467}$$

and $v_2 \equiv 2^{100} \equiv 189 \pmod{467}$. So $v_1 = v_2 = 189$.

Security of ElGamal Signatures

GMR-secure in the ROM assuming that H takes on random values and computing discrete logarithms modulo p is hard.

- Formally, one shows that the DLP reduces to existential forgery, *i.e.* that an algorithm for producing existential forgeries can be used to solve the DLP.

If Step 2 of the verification is omitted (verifying that $r < p$), a universal forgery attack is possible.

- More exactly, if an attacker intercepts a signature (r, s) to a message m, he can forge a signature (R, S) to an *arbitrary* message M.
- The resulting R satisfies $0 \le R \le p(p - 1)$.

Security of ElGamal Signatures, cont.

The public parameter g must be chosen verifiably at random (eg. publish PRNG, seed, and algorithm used) in order to ensure that g is a primitive root of p

If the same value of k is used to sign two messages, the private key x can be computed with high probability.

The Digital Signature Algorithm (DSA)

Invented by NIST in 1991 and adapted as the *Digital Signature Standard* (DSS) in Dec. 1994.

Variation of El Gamal signature scheme, with similar security characteristics, but much shorter signatures.

DSA Setup

A produces her public and private keys as follows:

1. Selects a 512-bit prime p and a 160-bit prime q such that $q \mid p-1$.
2. Selects a primitive root g of p.
3. Computes $h \equiv g^{(p-1)/q} \pmod{p}$, $0 < h < p$. Note that $h^q \equiv 1 \pmod{p}$ by Fermat's theorem, and if $a \equiv b \pmod{q}$, then $h^a \equiv h^b \pmod{p}$.
4. Randomly selects $x \in \mathbb{Z}$ with $0 < x < q$ and computes $y \equiv h^x \pmod{p}$

Public key: $\{p, q, h, y\}$ ($4 \cdot 512 = 2048$ bits)
Private key: $\{x\}$ (160 bits)

DSA also uses a cryptographically secure hash function $H : \{0,1\}^* \to \mathbb{Z}_q$. The DSS specifies that SHA-1 be used.

Signing and Verifying

A signs message $M \in \{0,1\}^*$ as follows:

1. Selects a random integer k with $0 < k < q$.
2. Computes $r \equiv \left(h^k \pmod{p}\right) \pmod{q}$, $0 < r < q$.
3. Solves $ks \equiv H(M) + xr \pmod{q}$. If $s = 0$, go back to step 1 (this happens with negligible probability).
4. A's signature is the pair $\{r, s\}$ (320 bits, as opposed to 1024)

B verifies A's signature as follows:

1. Obtains A's authentic public key $\{p, q, h, y\}$.
2. Computes the inverse $s^* \in \mathbb{Z}_q^*$ of $s \pmod{q}$.
3. Computes $u_1 \equiv H(M)s^* \pmod{q}$, $u_2 \equiv rs^* \pmod{q}$, and $v \equiv \left(h^{u_1} y^{u_2} \pmod{p}\right) \pmod{q}$, $0 < v < q$.
4. Accepts the signature (r, s) if and only if $v = r$.

Proof of Correctness

Proof of Correctness.

Note that $k \equiv (H(M) + x)s^* \pmod{q}$ and

$$\begin{aligned} v &\equiv h^{u_1} y^{u_2} \\ &\equiv h^{H(M)s^*} y^{rs^*} \\ &\equiv h^{H(M)s^*} h^{xrs^*} \\ &\equiv h^{(H(M)+xr)s^*} \\ &\equiv h^k \equiv r \pmod{p} \ . \end{aligned}$$

Now v and r are integers strictly between 0 and q that are congruent modulo the much larger modulus p. Hence $v = r$. □

Efficiency of DSA

Small signature (320 bits, much smaller than El Gamal) but the computations are done modulo a 512-bit prime.

Congruence in step 3 of signature generation has a "+" whereas the one in El Gamal has a "−".

The DSA verification procedure is more efficient than the way verification was described for ElGamal

- requires only two modular exponentiations in step 2 as opposed to three in ElGamal.

However, the one in ElGamal can be rewritten in the same efficient way

- check if $ry^{s^* r} \equiv g^{s^* H(M \| r)} \pmod{p}$ where s^* is the inverse of $s \pmod{p-1}$.

Parameter Sizes for Public-Key Cryptography

1024-bit RSA is estimated to provide 80 bits of security

- should be paired with a 160-bit hash function and an 80-bit block cipher (so that all three components equally strong).

Security levels and parameter/key sizes (NIST recommendations):

Security level (in bits)	80	112	128	192	256
Hash size (in bits)	160	224	256	384	512
RSA modulus (in bits)	1024	2048	3072	7680	15360

Security of DSA

Based on the belief that extracting discrete logs modulo q is hard (seems reasonable).

Proof of GMR-security does *not* hold, because $H(M)$ is signed as opposed to $H(M\|r)$ (reduction to DLP requires that the forger be forced to use the same r for two signatures)

More information: "Another look at provable security" by Koblitz and Menezes, *J. Cryptology* 2007; see "external links" page.