Perfect Forward Secrecy

18.095 Lecture 1

MIT

January 4, 2016

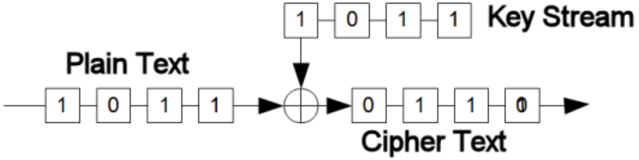
Secure communication over public networks

- Encryption: How can I communicate privately?
- Authentication: How can I know who I am really talking to?

One-time pad

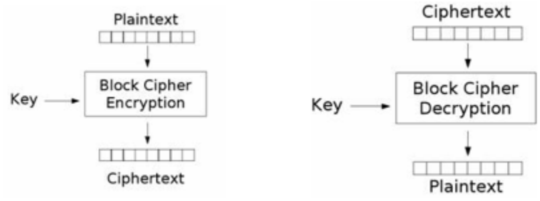
- Completely secure if used properly; acts as a random mapping.
- Extremely insecure if used improperly (e.g. as a two-times pad).
- Key distribution (and synchronization) is a big problem.

Stream ciphers



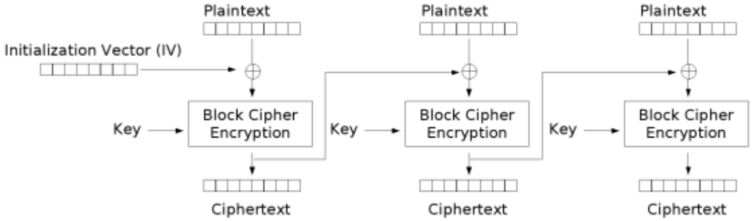
- Can be viewed as a generalization of a one-time pad.
- The key stream is typically a pseudo-random sequence generated from a (reasonably short) seed.
- The seed acts as a secret key.
- Security depends on how good the pseudo-random generator is.

Block ciphers



- Key specifies a bijective function (a permutation) from the set of all plaintext blocks to the set of all ciphertext blocks.
- Decryption uses the inverse function.
- Can be viewed as a generalization of a substitution cipher.
- Identical plaintext blocks always have identical ciphertext blocks; vulnerable to frequency analysis.

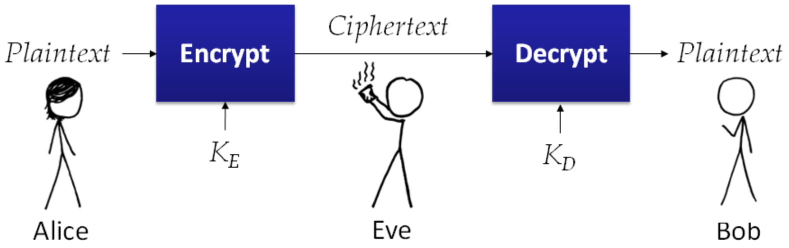
Cipher block chaining



Cipher Block Chaining (CBC) mode encryption

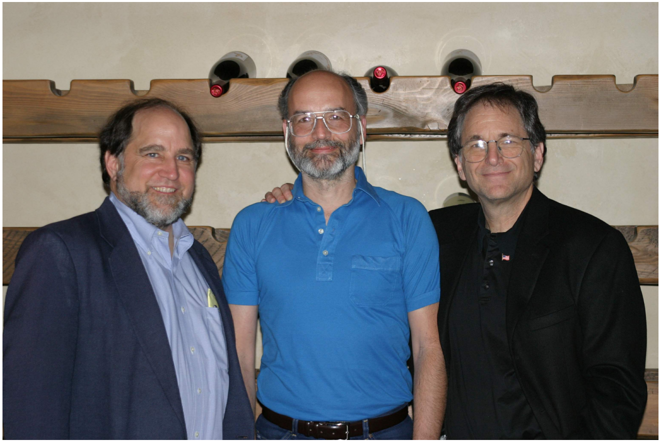
- Initialization vector is a random block that need not be secret.
- Chaining allows a block cipher to behave like a stream cipher.
- Identical plaintext blocks now have different ciphertext blocks.

Symmetric vs Asymmetric



- Symmetric encryption uses a shared secret key $K_E = K_D$.
- Asymmetric encryption has $K_E \neq K_D$.
- Typically the decryption key is secret, but the encryption key is not. Making K_E public solves the key distribution problem!

The RSA algorithm



Rivest Shamir Adleman

The RSA algorithm

- Let $n = pq$ be the product of two distinct primes.
- Let $r = (p-1)(q-1)$ and pick e relatively prime to r.
- Let d be the inverse of e modulo r; so $de = kr + 1$ for some k.
- Plaintext and ciphertext blocks are encoded as integers modulo n.

Encryption: $c \equiv m^e \bmod n$.
Decryption: $m \equiv c^d \bmod n$.

Claim: $m^{ed} \equiv m \bmod n$.

- By the CRT, it suffices to show $m^{ed} \equiv m \bmod p$ (and q).
- This is clearly true if $m \equiv 0 \bmod p$. Otherwise,

$$m^{ed} \equiv m^{kr+1} \equiv m^{k(p-1)(q-1)}m \equiv m \bmod p,$$

by Fermat's little theorem ($a^{p-1} \equiv 1 \bmod p$).

Fast exponentiation

To exponentiate efficiently, we may use binary exponentiation.

For example,

$$\begin{aligned}
x^{100} &= (x^{50})^2 \\
&= ((x^{25})^2)^2 \\
&= (((x \cdot x^{24})^2)^2)^2 \\
&= (((x \cdot x^{12})^2)^2)^2 \\
&= (((x \cdot (x^{6})^2)^2)^2)^2 \\
&= (((x \cdot ((x^{3})^2)^2)^2)^2)^2 \\
&= (((x \cdot ((x \cdot x^{2})^2)^2)^2)^2)^2.
\end{aligned}$$

To compute x^n requires at most $2\log_2 n$ multiplications.

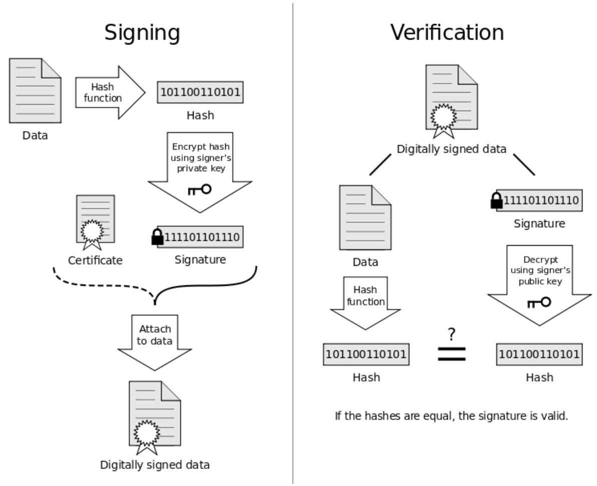
Security and speed of RSA

The exponent e is *public*, as is the modulus n (but not p or q).
The exponent d is *private*.

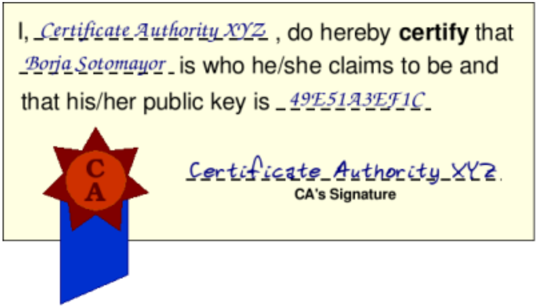
Anyone can encrypt a message using e, but without d decryption is believed to be as difficult as factoring n. When n is, say, a 2048 bit integer, factoring n is infeasible using any currently known algorithm.

Encryption, is reasonably fast, but not as fast as we would like.
RSA is typically used only for authentication and key exchange.
Faster symmetric algorithms are used for data encryption.

Digital signatures



Certificates


I, Certificate Authority XYZ , do hereby **certify** that Borja Sotomayor is who he/she claims to be and that his/her public key is 49E51A3EF1C

Certificate Authority XYZ

CA's Signature

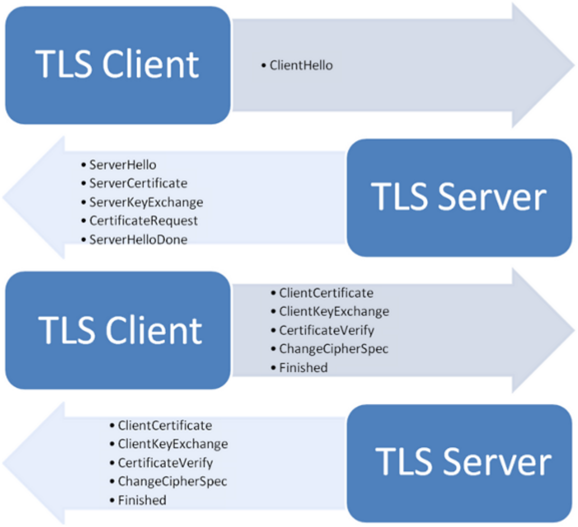
- Certificates solve the authentication problem.
- Any public-key algorithm can be used, RSA is typical.

Transport layer security (TLS)

A typical https session proceeds as follows:

1. Client contacts the server asking for a secure connection.
2. Server responds with a certificate, which includes its public key.
3. Client validates the server's certificate using a trusted CA.
4. Client generates a random session key, encrypts it with the server's public key, and sends it to the server.
5. Server decrypts the session key using its private key
6. The client/server session is now encrypted using the session key.

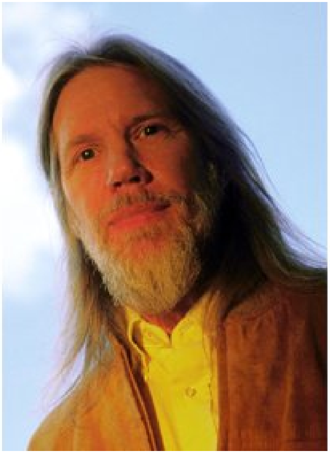
Transport layer security (TLS)

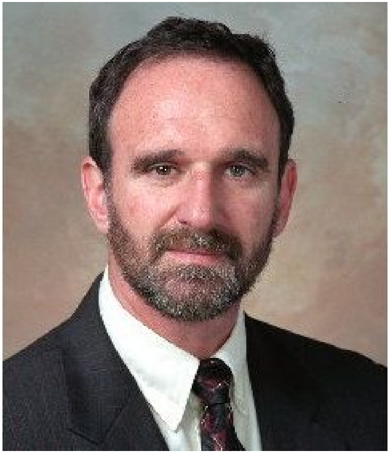


What could possibly go wrong...

- Good: Every session gets its own random key, so even if one session key is broken, the others remain secure.
- Bad: If the server's private key is broken, every session with this server (by any client, at any time in the past) can be decrypted.
- A patient adversary may collect encrypted session data in the hope of breaking the server's private key at some future date.
- Even 4096-bit RSA keys can be broken (by timing or gcd attacks). No cryptographic protocol can guarantee correct implementation.

Diffie-Hellman key exchange


Diffie


Hellman

Diffie-Hellman key exchange

Let p be a prime and let g be an integer reduced modulo p.
These may be published, or picked on the fly by Alice and sent to Bob.

1. Alice picks a random exponent a and sends $A = g^a \bmod p$ to Bob.
2. Bob picks a random exponent b and sends $B = g^b \bmod p$ to Alice.
3. Alice uses a and B to compute $S = B^a = (g^b)^a \bmod p$.
4. Bob uses b and A to compute $S = A^b = (g^a)^b \bmod p$.
5. The secret value S is now known to both Alice and Bob.

Eve may know p, g, A, B, but she does not know a or b, and if p is large (say 2048 bits), it is believed to be very hard to derive S from A and B.

Ephemeral Diffie-Hellman: The random integers a and b are not reused, they are discarded as soon as S is computed. The base g can easily be made ephemeral as well (but typically is not).

Discrete logarithms

The exponent a in $A = g^a \bmod p$ is called the *discrete logarithm* of A (with respect to g), and may be written as

$$a = \log_g A$$

If we take a to be the least positive integer for which $A \equiv g^a \bmod p$ holds, then a is uniquely defined.

If Eve can compute discrete logarithms, she can recover S from A or B. This is believed to be hard, in general (roughly as difficult as factoring).

It is also believed that breaking Diffie-Hellman is as hard as computing discrete logarithms (but neither of these beliefs has been proved).

Groups

A group is a pair $(G, \star)$ where $\star$ denotes a binary operation that maps pairs of elements of G to another element of G (typically $\star$ is $+$ or $\times$).

The pair $(G, \star)$ must satisfy the following

1. There is an *identity* $i \in G$ such that $i \star g = g \star i = g$ for all $g \in G$.
2. Every $g \in G$ has an *inverse* $h \in G$ such that $g \star h = h \star g = i$.
3. $\star$ is *associative*: $(a \star b) \star c = a \star (b \star c)$ for all $a, b, c \in G$.

Examples:

- The set of integers $\mathbb{Z}$ form an infinite group under addition.
- The set $\mathbb{Z}/n\mathbb{Z}$ (residue classes mod n) is an additive group.
- The nonzero elements of $\mathbb{Z}/p\mathbb{Z}$ form a multiplicative group.

All these groups are *cyclic*; they can be generated by a single element.

Discrete logarithms in finite groups

The Diffie-Hellman protocol and the notion of discrete logarithms make sense in any finite group G, even when the group operation is not $\times$.

But not all groups are created equal, even those of the same size. The difficulty of computing discrete logarithms depends on the group.

Example: Let p be a prime.
The multiplicative group $(\mathbb{Z}/p\mathbb{Z}, \times)$ is a cyclic group of order $n = p - 1$.
Thus it is isomorphic to the additive group $(\mathbb{Z}/n\mathbb{Z}, +)$.

In $(\mathbb{Z}/n\mathbb{Z}, +)$, computing "discrete logarithms" corresponds to division, which is very easy (using the extended Euclidean algorithm).

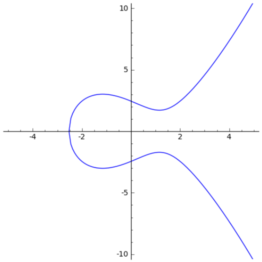
So it is important to pick the right group!

Elliptic curves

An *elliptic curve* is defined by an equation of the form

$$y^2 = x^3 + Ax + B$$

with $4A^3 + 27B^2 \neq 0$. For example,



$$y^2 = x^3 - 4x + 6$$

The projective plane

Over any field $\mathbb{F}$, the projective plane consists of all triples $(x : y : z)$, with $x, y, z \in \mathbb{F}$ not all zero, subject to the equivalence relation

$$(x : y : z) \sim (\lambda x : \lambda y : \lambda z) \qquad \text{(for all } \lambda \neq 0\text{).}$$

Given an equation for an elliptic curve, we *homogenize* it as

$$zy^2 = x^3 + Axz^2 + Bz^3$$

and consider projective solutions $(x : y : z)$; these are *points* on E.

Each point $P = (x : y : z)$ with $z \neq 0$ can be written as $(x : y : 1)$, corresponding to an *affine* point (x, y) in the Cartesian plane.

There is exactly one point on E with $z = 0$, the *point at infinity*:

$$\infty = (0 : 1 : 0).$$

Lines and elliptic curves

Suppose P and Q are two projective points on an elliptic curve E. They determine a line L (if $P = Q$ then L is the tangent line at P).[1]

A fundamental result of projective geometry known as Bezout's Theorem implies that L must intersect E at a third point R.

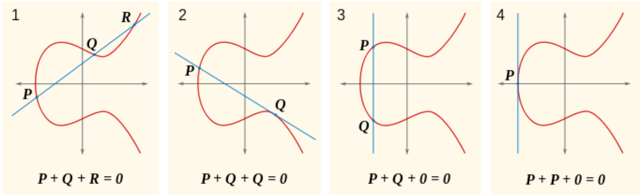
If L is tangent to P but not Q, then the intersection at P is considered a double intersection. In this case $R = P$.

If $P = Q = \infty$, the tangent line L has a triple intersection at P and we have $R = \infty$ (all other tangents are double intersections).

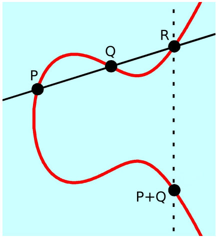
[1] Requiring $4A^3 + 27B^2 \neq 0$ ensures there is a unique tangent line.

The elliptic curve group law

Three points on a line sum to zero.



Zero is the point at infinity.



The elliptic curve group law

With addition defined as above, the set of rational points on an elliptic curve forms a commutative group.

- The point $(0:1:0)$ at infinity is the identity element 0.
- The inverse of $P = (x:y:z)$ is the point $-P = (x:-y:z)$.
- Commutativity is obvious: $P+Q = Q+P$.
- Associativity is not so obvious: $P+(Q+R) = (P+Q)+R$.

The computation of $P+Q=R$ is purely algebraic. The coordinates of R are rational functions of the coordinates of P and Q, and can be computed in any field.

We are interested in finite fields $\mathbb{F}_p$; for a prime p this corresponds to $\mathbb{Z}/p\mathbb{Z}$, whose elements we may represent as integers in $[0, p-1]$.

The elliptic curve discrete logarithm problem

By adding a point to itself repeatedly, we can compute $2P = P + P$, $3P = P + P + P$, and in general, $nP = P + \cdots + P$ for any positive n.

Given points P and $Q = nP$, the *discrete logarithm problem* asks for n.

If the cyclic group generated by P has prime order N, the fastest known algorithm for solving this problem requires at least $\sqrt{N}$ operations.

Thus if N is a 256-bit prime we achieve a 128-bit security level.

To achieve the same level of security using the discrete logarithm problem in a finite field (or RSA), we would need a 3072-bit modulus.

This makes elliptic curves very attractive for cryptographic protocols based on the discrete logarithm problem.

ECDHE_RSA summary

Setup:

- Server creates an RSA key pair and obtains a certificate.
- Elliptic curve over finite field with a large subgroup $\langle G \rangle$ of prime order p generated; typically standardized.

Protocol:

- Client generates random $a \in \mathbb{Z}/p\mathbb{Z}$ and sends aG to server.
- Server generates random $b \in \mathbb{Z}/p\mathbb{Z}$, computes $S = abG$, sends signed bG to client (discards b).
- Client verifies signed bG, computes $S = abG$ (discards a).
- Client and server use S as a secret key for a symmetric encryption for the current session (when session ends, both discard S).

Key point:

- Leaking a, b, S does not compromise any other sessions.