

Asymmetric Cryptography

Hash Functions

Security Engineering and Failure Modes

Aram Khalili

Department of Computer Science

University of Maryland

Asymmetric Cryptography

- First publicly suggested in 1976 by Diffie & Hellman.
- Called asymmetric cryptography because *different* (non-symmetric) keys are needed for encryption and decryption. Also called public-key cryptography, because one of the keys is made public.
- Diffie & Hellman found a one-way function, a function that is easy to compute but hard to invert, in the discrete logarithm $a^x = b \pmod p$.
- Given a , x and p it is easy to find b . But given a , a^x , b and p it is comparatively hard to find x , and becomes computationally infeasible for sufficiently large numbers.

DH key agreement

- From this problem Diffie & Hellman built a key agreement protocol:
 - Alice and Bob want to establish a common secret over an insecure channel.
 - Let large prime p and generator α be public.
 - Alice secretly picks a random number A , computes $\alpha^A \bmod p$, and sends it to Bob.
 - Bob secretly picks a random number B , computes $\alpha^B \bmod p$, and sends it to Alice.
 - Each raises the received number to their secret exponent and computes $\alpha^{AB} \bmod p$. This is their shared secret.
- Many call this the first public-key cryptosystem, but it's not. It's a key agreement/establishment protocol.

RSA

- RSA (Rivest, Shamir, Adleman) was the first (1978) published public-key crypto- and digital signature system. It is based on exponentiations, discrete logarithms and factoring:
 - Alice wants to send Bob a secret message M . She looks up Bob's public key e, n . ($M < n$)
 - She computes $M^e \bmod n = C$ and sends it to Bob.
 - Bob uses his private key d to decrypt and computes $C^d \bmod n = M$.
 - Only Bob can decrypt C because only he knows d , but anyone can encrypt, because e and n are public.
- RSA, as all public-key cryptosystems, is based on a trapdoor one-way function, a function that is hard to invert unless you have some special knowledge.

Why does this work?

- Because of the relation of e , d and n and Euler's theorem.
- Euler's theorem is a generalization of Fermat's Little Theorem.
 - If $\gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.
 - $\phi(x)$ is Euler's Totient function, defined to be the number of positive integers less than and relatively prime to x .
 - Relatively prime means sharing no common prime factors, or $\gcd(a, b) = 1$.
 - Since $n = p \cdot q$, where p and q are large primes, $\phi(n) = (p - 1)(q - 1)$.
 - The relation between e and d is that $e \cdot d \equiv 1 \pmod{\phi(n)}$, such that $M^{ed} \equiv M^1 \pmod{n}$.

RSA signatures

- RSA messages are constructed such that everybody with Bob's public key can encrypt, but only Bob decrypt.
- RSA signatures are constructed such that only Bob can sign, but everyone with Bob's public key can verify:
 - To sign a message M , Bob raises it to his private exponent d :
 $M^d \bmod n = S$.
 - To verify, anyone with the public key raises the signature to the public exponent e : $S^e \equiv M^{ed} \equiv M^1 \bmod n$.
- For efficiency and performance reasons a fixed length message digest is often signed instead of the full message itself.

Other public-key systems

- RSA was the first system, and it is simple to understand and implement, however, in some cases, a message can be recovered without knowing the secret key, and an encryption or signature of the same message will always be the same.
- The ElGamal cryptosystem incorporates a random number in each in encryption or signature, so that even if a document is encrypted or signed twice, the ciphertext/signature need not be the same. The ElGamal system is based on the discrete logarithm problem.
- The NTRU cryptosystem was developed because its designer think that it is more efficient than RSA. NTRU also includes a random number and is based on polynomial rings.
- The McEliece system is based on algebraic coding theory (error-correcting codes).

Bishop's Criteria for public key systems

- It must be computationally easy to encipher or decipher a message given the appropriate key.
- It must be computationally infeasible to derive the private key from the public key.
- It must be computationally infeasible to determine the private key from a chosen plaintext attack.

Cryptographic Checksums

- Cryptographic checksums or message digests are hash functions. As such they are projection functions, and map a variable input length to a fixed output size. Hash functions (& projection functions in general) are one-way functions.
- In order for it to be cryptographically strong and useful, hash function h needs to fulfil the following criteria:
 - h must be easy to compute.
 - h must be hard to invert, i.e. given $h(x)$ is infeasible to find x . (Called *pre-image resistant*.)
 - Given x and $h(x)$ it must be computationally infeasible to find any $x' \neq x$ such that $h(x) = h(x')$. (Called *collision intractable*.)
 - It must be infeasible to find $x, x', x \neq x'$ such that $h(x) = h(x')$. (Called *collision resistant*.)

Usage

- (Cryptographic) hash functions are usually used to reduce data for authentication protocols. They are supposed to associate each message uniquely to a hash value.
- Hash function are assumed to be publicly known.
- The hash value can then be signed or authenticated instead of the full message. If the signature/authentication and hash mapping hold, the message can be assumed to be authentic.
- There are 2 forms of authentication with cryptographic checksums, symmetric and asymmetric based ones.

Asymmetric authentication

- Asymmetric authentication are signatures, and authenticate one principal to anyone with the public key.
- To authenticate a message M , first hash it: $h(M) = H$, then sign the hash value H with the private key d : $H^d \bmod n = C$.
- To verify, the receiver exponentiates the signature with the public exponent to get H . Then he hashes the message M and compares the results. If they match, the verification succeeded.

Symmetric authentication

- Symmetric authentication uses secret keys and authenticates among the group of principals that know the particular key.
- Authentication is done with keyed hash functions, or HMACs.
- Instead of hashing just the message, the secret key is pre- and appended to it. The hash can then be added to the message as an authentication code.
- The receiver, who also knows the secret key, also appends and prepends the key to the message, then compares the results to the authentication code. If they match, the authentication succeeded.

Security Engineering

- Ross Anderson's *Why cryptosystems fail* is an excellent survey of failure modes of security engineering.
- It mentions the secrecy that surrounds most organizations' security setup and the unwillingness to allow security incidents to become public.
- This causes a lack of feedback and denies the opportunity to learn from mistakes.
- Security through obscurity in general not a good idea.

Bank ATMs

- Anderson chose bank ATMs as an example because their faults have some observability, particularly in courts, and probably also because of some personal knowledge and involvement with banking security procedures.
- Also, the paper is aimed at a general audience, to which bank ATMs have more relevance and familiarity than many other applications of security engineering.
- Anderson's title focuses on cryptographic systems, however, in his examples many other security aspects are involved such as access control or separation of duty.

Failure Modes

- Anderson goes through a series of examples illustrating the lack of proper security engineering in ATM setup and ATM card/PIN procedures.
- Among the failure modes are
 - improper access control
 - improper authentication
 - improper separation of duty
 - improper training/incompetence (user failure)
 - faulty design
 - incomplete threat analysis/threat model

Causes

- Anderson identifies 2 main causes of these failures.
- One is that the threat model is wrong.
- The other is the wrong assumption that system implementors, administrators and users are knowledgeable enough to design, implement, maintain and use the system properly.

Solutions?

- Anderson then argues for a paradigm shift from security as a black-box “plug-in” to a process, and makes several recommendations to ensure that this process is carried out adequately.