Crypto Overview Note

Trent Jaeger
Systems and Internet Infrastructure Security Laboratory
The Pennsylvania State University, University Park PA 16802

In this note, we provide basic definitions and outline some fundamental cryptographic constructions.

Nomenclature Here are the set of primitive crypto statements:

- $a, b, c, ...$ are data
- $A, B, C, ...$ are principals (A=Alice, B=Bob, C=Charlie, ...)
- PW^A is the password of Alice (A)
- K is a key
- n is a nonce, a random number used in establishing the freshness of a communication
- K^{AB} is a key shared between Alice and Bob (symmetric key cryptosystem)
- K_A^+ and K_A^- are Alice's public and private keys, respectively (public key cryptosystem)
- $E(d, K)$ is an encryption of data d with key K (K^{AB} for symmetric key and K_A^+ or K_A^- for public and private key encryption by Alice)
- $H(d)$ is the cryptographic hash of data d
- $HMAC(d, K)$ is a HMAC message authentication code used to confirm integrity in symmetric key cryptosystems
- $S(K_A^-, d)$ is Alice's signature of data d ($S(K_A^-, d) = E(K_A^-, H(d))$)
- '+' or '—' refer to concatenation

Cryptographic operations aim to protect the secrecy of communication (Secrecy), verify that the message has not been tampered (Integrity), and enable identification of the source of message (Authenticity).

Secrecy To protect the secrecy of data, we encrypt the data $c = E(d, K_{AB})$, where c is the ciphertext. *Symmetric key encryption* limits access to the holders of the key. K_{AB} can be read by both A and B, but no other parties (assuming proper key distribution). Decryption uses the same key, $d = D(c, K_{AB})$.

Thus, for Alice to send a secret message to Bob and for Bob to decrypt and read it, we have:

- Alice and Bob establish a symmetric key K_{AB}
- Alice encrypts: $c = E(d, K_{AB})$
- Bob decrypts: $d = D(c, K_{AB})$
- Roles are reversed when Bob sends a secret message to Alice

Using *public key cryptography*, a public key encryption is used to protect secrecy. If anyone wants to send a secret message to Alice they generate $c = E(d, K_A^+)$. Note that this operation can be performed by any party, since the public key is available to all. Only Alice can decrypt the resultant ciphertext c because only she has the corresponding private key, K_A^-.

- Alice has a private key, K_A^-
- Anyone can obtain the corresponding public key K_A^+
- Anyone can encrypt a message to Alice: $c = E(d, K_A^+)$
- But, only Alice can decrypt the message: $d = D(c, K_A^-)$

Integrity Simply encrypting a message does not ensure that the receiver can determine whether a received message was the original encrypted message. This is due to the mechanisms used to encrypt messages, called *modes*, and because the content of the message may make it hard for a receiver to determine if its has been tampered.

Suppose Alice sends a random number to Bob, and she encrypts it to keep it secret. Mallory (the malicious adversary) may modify the encrypted message, and Bob will not be able to tell whether this was the original that Alice sent, because he doesn't know what the intended plaintext value was.

In symmetric key cryptography, we use a *message authentication code* (MAC) to enable a remote party to verify the integrity of an encrypted message. A MAC is a keyed-hash, $mac = H(d|K_{AB})$ using the shared key of the two parties.

An HMAC is a specific MAC function, resistant against some attacks that are possible against the naive MAC function above, $HMAC(d, K_{AB}) = H(K_{AB}|h(K_{AB}|d))$. Use HMAC to protect integrity of symmetric key encrypted messages.

To send a secret, integrity-verifiable message using a symmetric key cryptosystem:

- Alice and Bob establish a symmetric key K_{AB}
- Alice encrypts and MAC's her message for integrity: $c = E(d, K_{AB})|h = HMAC(d, K_{AB})$
- Bob decrypts: $d = D(c, K_{AB})$

- Bob verifies the integrity of decrypted message: $h' = HMAC(d, K_{AB}) == h$?
- Roles are reversed when Bob sends a secret message to Alice

Using public key cryptography, signatures are used to protect integrity. A signature is a private key encryption of a hash of the message, $s = E(H(d), K_A^-)$. Note that a signature encryption does not provide secrecy (any can decrypt with the corresponding public key), but only one principal could have generated the signature. The hash ensures that changes to the data do not go undetected, as in the MAC case.

To send a secret, integrity-verifiable message using a public key cryptosystem:

- Alice has a private key, K_A^-
- Anyone can obtain the corresponding public key K_A^+
- Bob has a private key, K_B^-
- Anyone can obtain the corresponding public key K_B^+
- Bob encrypts a message to Alice: $c = E(d, K_A^+)|s = E(H(d), K_B^-)$
- Only Alice can decrypt the message: $d = D(c, K_A^-)$
- She can also verify that the message is really what Bob sent: $s' = D(H(d), K_B^+) == s$?

We also write signatures in the shorter way, $S(d, K_A^-) = E(H(d), K_A^-)$.

Authenticity Authenticity is the ability to determine who really generated a particular message.

For symmetric key cryptography, any party that holds the key may have generated a message. For example, a message encrypted by K_{AB}, such as $E(d, K_{AB})$ could have been generated by either Alice or Bob. While Alice probably knows when Bob generated a message instead of her (if the integrity is protected via an HMAC), it is not possible for her to prove to a third party that only Bob could have generated the message, as both have the capability (i.e., the key).

The problem of proving to a third party that a message was generated by a specific principal is the *non-repudiation problem* because that principal cannot deny generating the message without being detected. It cannot be solved in symmetric key cryptosystems, but it can in public key cryptosystems.

For a public key system, the signature of a message is a non-repudiable statement that the message was generated by the signing party. Thus, you must take care in signing messages. For example, you should not sign an encrypted message because it may say something that you don't approve of.

Key Distribution The final problem is to get the keys for symmetric and public key systems in the first place. The goal is to obtain a key that enables verification of the secrecy and integrity of another principal's messages. That is, we want to determine that a key is shared only with a specific principal (for symmetric key cryptosystems) or that only a specific principal has private key corresponding to a particular public key (for public key cryptosystems). This problem is called the *key distribution problem.*

Needham-Schroeder provides protocol designs for key distribution in both kinds of systems. For symmetric key systems, the challenge is to prove *freshness* of a particular distributed key with a remote principal (supported by a *trusted third party*). A *nonce* is a (pseudo)random value used to provide freshness.

For public key systems, the challenge is to verify that the mapping between a private-public key pair obtain from a trusted third party (or via a certificate signed by a special trusted third party, a *certificate authority*). Private keys may be exposed or cracked, so a prior statement from a trusted third party may not longer be accurate. *There is actually a bug in the Needham-Schroeder public key protocol, it must not be used.*

A certificate binding a principal to a public key from a certificate authority CA looks like this:

- CA has a private key, K_{CA}^{-}
- The CA's public key is widely broadcast, so everyone knows it K_{CA}^{+}
- CA generates a certificate $cert_A = K_A^{-}|data$ for Alice's public key, $S(cert, K_{CA}^{-})$
- Remote principals who obtain Alice's certificate can verify that CA certified it via a signature verification

Often, you will use a public key cryptosystem to support a key distribution protocol for a symmetric cryptosystem. That is, we use public key certificates to help identify parties and distribute a symmetric key for subsequent use (see SSL/TLS).

Summary Typically, all of these tasks, key distribution, secrecy, integrity and authenticity, must be met in order to complete a secure communication.

You must know these basics, when to apply them, and how to combine into effective cryptographic constructions.