Cryptography: Purposes

- Source: Ross Anderson,
  Security Engineering.

Cryptography—the study of the design of ciphers—is a tool used to help meet several goals, among them:

- Privacy: others can't read our messages.
- Integrity: others can't change our messages without us knowing.
- Authentication: we know whom we're talking to.

Some common terminology: we convert from plaintext to ciphertext and back (decryption).

- Although we typically think of text messages as characters, our algorithms generally process streams of numbers or bits, making use of standard encodings of characters as numbers.

Substitution

- Simplest scheme is just to permute the alphabet:
  \[
  \text{tylerduniabcfghjkmopqsvwxz} \]

- So that "so long and thanks for all the fish" \[\Rightarrow\] "ohtchgutygrtpnygbotdhmtycctpn\tdion"

- Problem: If we intercept ciphertext for which we know the plaintext (e.g., we know a message ends with name of the sender), we learn part of the code.

- Even if we have only ciphertext, we can guess encoding from letter frequencies.

Stream Ciphers

- Idea: Use a different encoding for each character position. Enigma was one example.

- Extreme case is the One-Time Pad:
  Receiver and sender share random key sequence at least as long as all data sent. Each character of the key specifies an unpredictable substitution cipher.

- Example:
  Messages: attack\at\dawn\|oops cancel that order\|attack is back on
  Key: vnchkjskruwisn\|tjcdktjdjsahtjkdhjrizn\|akjqltpot pfhsdjrsqieha...
  Cipher: vfvhmtrkjtzin|gxrvjvjqlwlglqkwgxhlcd|acbqnco wkoghuniee

- Unbreakable, but requires lots of shared key information.

- Integrity problems: If I know message is "Pay to Paul N. Hilfinger $100.00" can alter it to "Pay to Paul N. Hilfinger $999.00" [How?]

Aside: A Simple Reversible Combination

- The cipher in the last slide essentially used addition modulo alphabet size as the way to combine plaintext with a key.

- Usually, we use a different method of combining streams: exclusive or (xor), which is the "not equal" operations on bits, defined on individual bits by
  \[
  x \oplus y = \begin{cases} 
  0 & \text{if } x \text{ and } y \text{ are the same}, \\
  1 & \text{else}
  \end{cases}
  \]

- Fact: \[x \oplus y \oplus x = y\]. So,
  \[
  01100011 11010110
  \oplus 10110101
  \oplus 10110101
  11010110 01100011
  \]

- In Python, and Java, this operation is written \[x ^ y\].

Using Random-Number Generators

- Python provides a pseudo-random number generator (used for the Hog project, e.g.): from an initial value, produces any number of "random-looking" numbers.

- Consider a function that creates pseudo-random number generators:

- Python provides a pseudo-random number generator (used for the Hog project, e.g.):
  \[
  \text{import random; def bitstream(seed): r = random.Random(seed); return lambda: r.getrandbits(1)}
  \]

- Using two sides of a conversation share the same key to use as a seed, can create the same approximation to a one-time pad, and thus communicate secretly.
Hello, world

### Signatures

#### RSA Public-key Encryption

The data encryption standard (DES) used the strategy with 12 rounds enough.

#### Block Ciphers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| Feistel Ciphers | A strategy for generating block ciphers. Break messages into two halves, and each half is transformed into another.
| DES      | A block cipher that uses the Feistel structure. It has 16 rounds, and the key is 64 bits long. |
| AES      | A symmetric key block cipher that is widely used. It has 128, 192, or 256-bit key sizes and 12 rounds. |

#### The Data Encryption Standard (DES)

- Uses a structure similar to Feistel ciphers.
- Consists of 16 rounds, with each round involving substitution and permutation operations.
- The key schedule is a complex and round-specific process that generates intermediate subkeys from the original key.
- The algorithm is described using a sequence of steps that transform the plaintext into the ciphertext.

#### Public Key Cryptography

- Use public-key cryptography to encrypt messages with a public key.
- The public key is known by everyone, while the private key is kept secret.
- When encrypting a message, it is transformed using the public key, which cannot be reversed.
- Decryption requires the private key, which is never shared.

#### RSA Encryption: The Math

- To encrypt a message, use the public key $(n, e)$ to compute $c = m^e \mod n$.
- To decrypt a message, use the private key $(d)$ to compute $m = c^d \mod n$.
- The values of $n$ and $e$ are chosen such that $ed \equiv 1 \pmod{\lambda(n)}$.

#### Example

- Suppose we want to encrypt the message $m = 5$.
- Choose a public key $(n, e) = (11, 3)$.
- The modulus $n = pq$ is chosen to be 11.
- Compute $c = 5^3 \mod 11 = 125 \mod 11 = 3$.
- The ciphertext $c = 3$ is then sent to the recipient.

#### Feistel Ciphers

- Used in the Data Encryption Standard (DES), among other applications.
- Consist of multiple rounds, each involving a permutation and a substitution step.
- The output of each round is fed back into the next round, allowing for a chaining effect.

#### Theoretical Conclusion

- The use of public-key cryptography, especially RSA, offers significant advantages over symmetric-key cryptography.
- It enables secure communication over insecure channels without the need for a shared secret key.
- However, it also introduces computational overhead compared to symmetric-key methods.

---

**If you have any questions or need further clarification, feel free to ask!**
Authentication on the Web

• When you talk to Amazon, how do you know that the exchange is private and that it really is Amazon you are talking to? (On the Internet, nobody knows you are a dog.)

• Certain companies (such as Verisign) issue signed certificates that say (in effect) "Verisign certifies that Amazon has public key X.

• Your browser comes equipped with Verisign's public key so you can verify their signature.

• At that point, you know how to talk to Amazon in a way that only they can understand.

• (As usual, the actual protocol is rather more complex.)

Playing Cards Over the Phone

• How do I play a card game over the phone, so that neither side can (undetectably) cheat?

• To keep it simple, assume we have a two-person game between Alice and Bob where all cards get revealed.

• For each game, let each side choose a secret encryption key, and assume an algorithm that is commutative: if a message is encrypted first by secret key A and then by key B, it can be decrypted by the two keys in either order.

• If I choose to go left or right at random, and Bob selects an edge from the chosen end, Alice can decrypt the message by applying the two encryption keys in random order.

• Alice deals cards to Bob by selecting and decrypting them, and sending them to Bob, who can decrypt them.

• Alice deals cards to herself by sending them to Bob, having him decrypt them and send them (now singly encrypted) back to Alice.

• At the end of the game, all information can be revealed, and both sides can check for consistency.

Extra: Zero-Knowledge Proofs

• Zero-Knowledge Proofs involve another kind of keeping information hidden even as one communicates certain characteristics of it.

• Suppose I possess the answer to a puzzle, and want to convince you that I have the answer without revealing anything about what it is.

• This is an example of a zero-knowledge proof (Abadi, Goldwasser, and Rackoff).

• Many uses, such as authentication (I want to prove who I am), or enforcing honesty while maintaining privacy.

Example (from Jean-Jacques Quisquater via Wikipedia): Peggy wishes to prove to Victor that she knows the password to get through the hidden door.

1. Peggy chooses to go left or right at random, without Victor seeing.

2. Victor then shouts out which side he wants her to come out.

3. Peggy uses her knowledge or not as necessary to emerge from the desired side.

• After several rounds, Victor is convinced that Peggy knows the password, but doesn't know it himself.

• An observer doesn't know if Peggy and Victor are colluding, and so does not learn that Peggy knows the password.

Illustration

• (as usual, the actual protocol is rather more complex)

• They can incorporate:

• If I'm the owner, you know how to talk to Amazon in a way that only you can understand.

• If I'm an employee, you can use the Amazon public key to sign messages.

• Then, I can use the Amazon public key to sign messages.

• You, for your part, can verify that the signature is from Amazon, even though you can't read the contents of the message (so long as you're using proper tools and techniques).

• Intuitively: I show you a dog's pawprint, and you're happy you're talking to Amazon.

• When you talk to Amazon, how do you know that the exchange is authentic?