Encryption Basics

ECE @ UT

Symmetric Encryption: Shift cipher

- **Shift-by-K**
  - Caesar supposedly used shift-by-3
- (current-symbol + K) mod alphabet-size
  - ***Stream cipher*** with key k,k,k,k,k,…
- Easy to break: N guesses for K
  - Also, statistics preserving encryption. Word length, letter frequencies.
  - External knowledge of letter frequencies
  - Chosen plaintext attack

Substitution Cipher

- **Key is a permutation of the entire alphabet**
  - More keys than shift cipher
  - With 26 letters, 26! Keys. (2^{88})
  - Sherlock Holmes, Adventure of the Dancing Men

- **Statistical attacks**
  - Letter frequencies. Combine with bi-, tri-grams.
  - Plain text letter always maps to same cipher-text letter: Mono-alphabetic cipher.

Poly-alphabetic Substitution Cipher

- **Use multiple substitution keys**
  - Example: key for odd and even letters.

Plaintext alphabet	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ciphertext alphabet one	TMKGOYDSIPELUAVCRJWXZNHBQF
Ciphertext alphabet two	DCBAHGFEMLKJIZYXWVUTSRQPON

- **Key search space for attacker: $(26!)^2$**
  - Key size: 26 x 2. How to remember? Share?

- Vigenere Cipher: **each sub-key restricted to a shift operation. Key size: 2 digits.**
  - Stream cipher with key stream k1.k2.k1.k2…
  - Length of keyword known → easy to break

Permutation Ciphers

- **Permute the letters in a *block***
  - Break text into block, pad its length, apply permutation
- **Weaknesses: statistics attacks and chosen plain-text attacks**.
  - Length of block

Definitions

(Perfect Secrecy). *A cryptosystem has perfect secrecy if*

$$p(P = m | C = c) = p(P = m)$$

for all plaintexts m *and all ciphertexts* c.

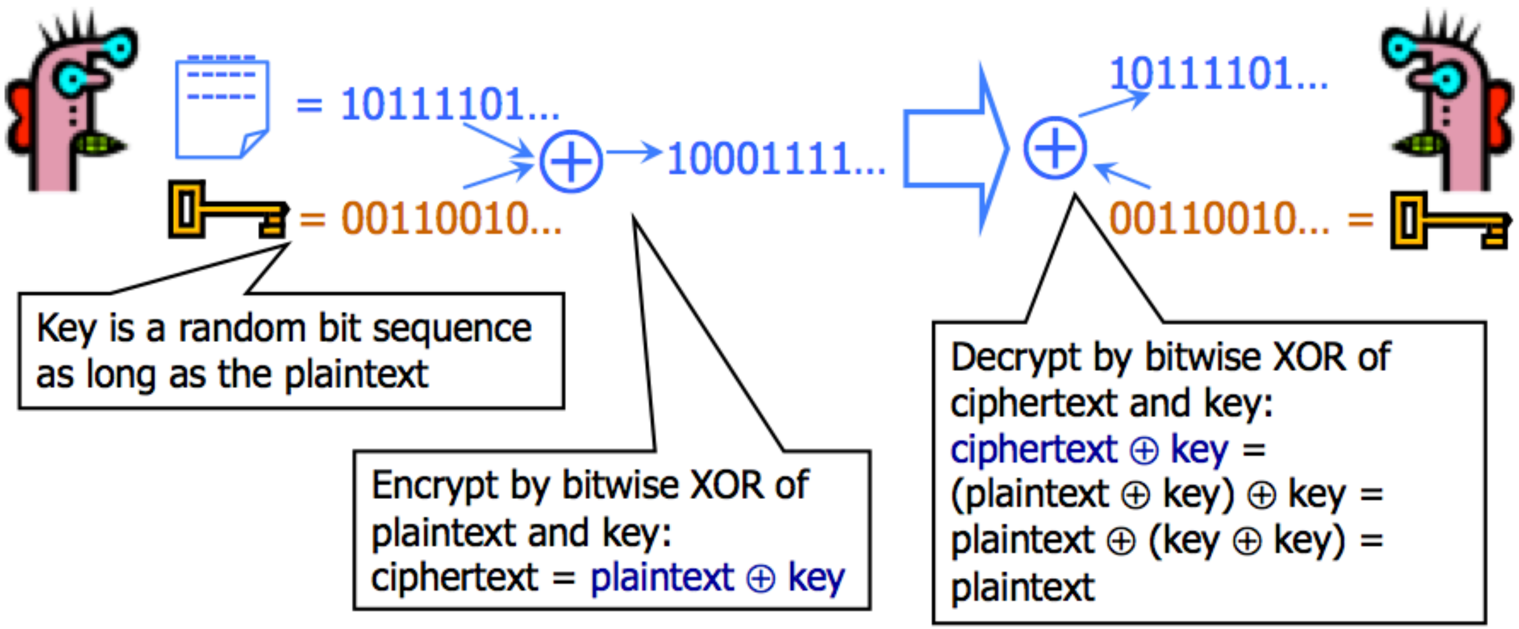
$$(\mathbb{P}, \mathbb{C}, \mathbb{K}, e_k(\cdot), d_k(\cdot))$$

denote a cryptosystem with $\#\mathbb{P} = \#\mathbb{C} = \#\mathbb{K}$. *Then the cryptosystem provides perfect secrecy if and only if*

- *every key is used with equal probability* $1/\#\mathbb{K}$,
- *for each* $m \in \mathbb{P}$ *and* $c \in \mathbb{C}$ *there is a unique key* k *such that* $e_k(m) = c$.

Perfectly Secure Cipher

- One-time Pad. [pc: Shmatikov]



- Easy to compute.

One Time Pad: Weaknesses

- ?
- Key sequence has to perfectly random
  - How?
- Does not guarantee integrity
  - Change plaintext to desired value.
- Keys should not be reused.

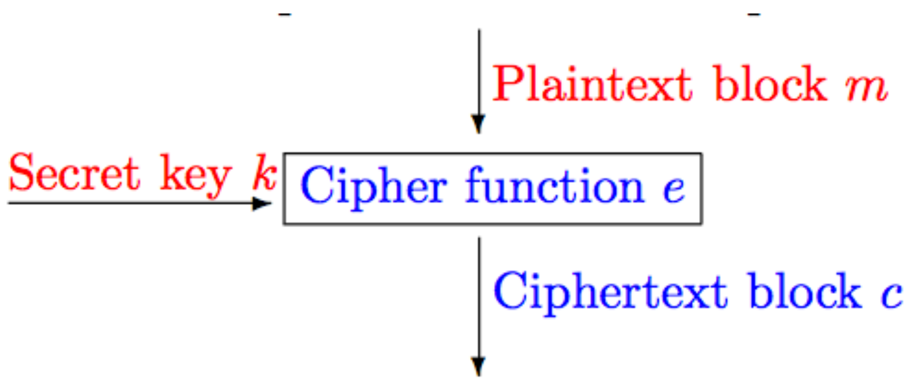
Learn relationship between plaintexts

$C1 \oplus C2 = (P1 \oplus K) \oplus (P2 \oplus K) =$

$(P1 \oplus P2) \oplus (K \oplus K) = P1 \oplus P2$

Block Ciphers

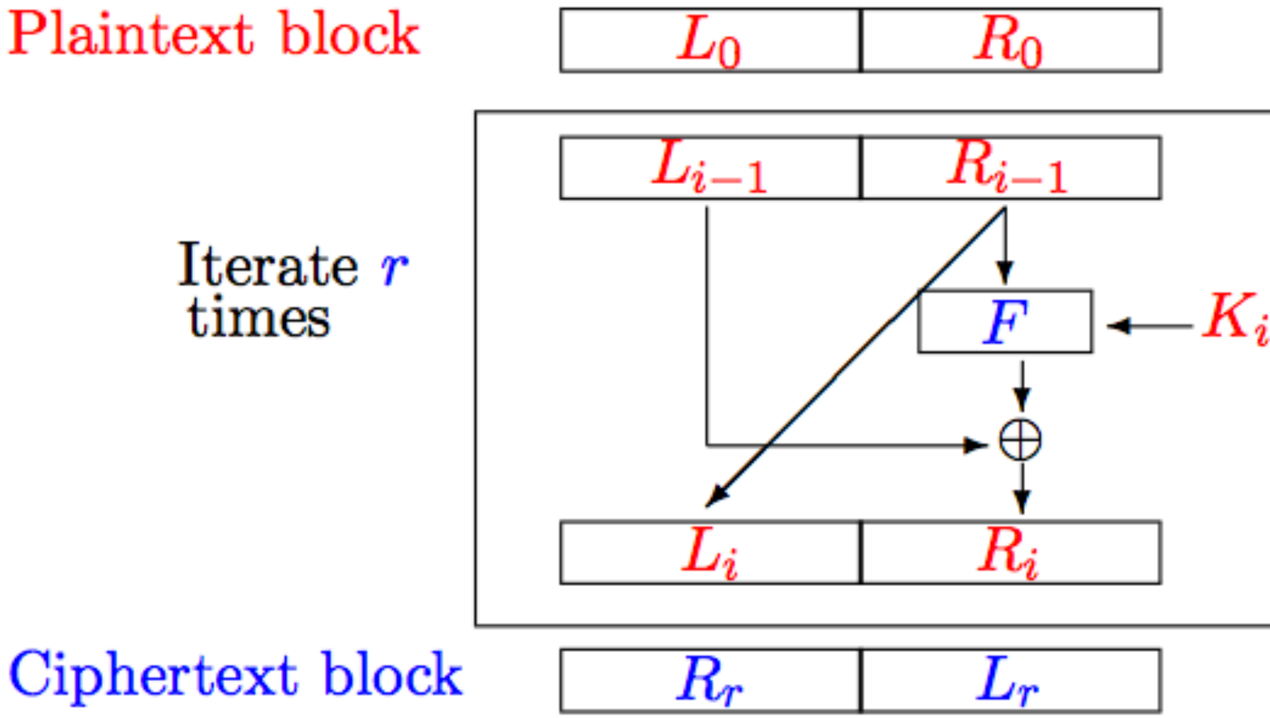
- Reduce key size. But also lose ‘perfect’ secrecy.



- 64b DES, 128b AES
- For long messages, **modes** of operation
  - ECB, CBC, Counter, …

Feistel Ciphers and DES

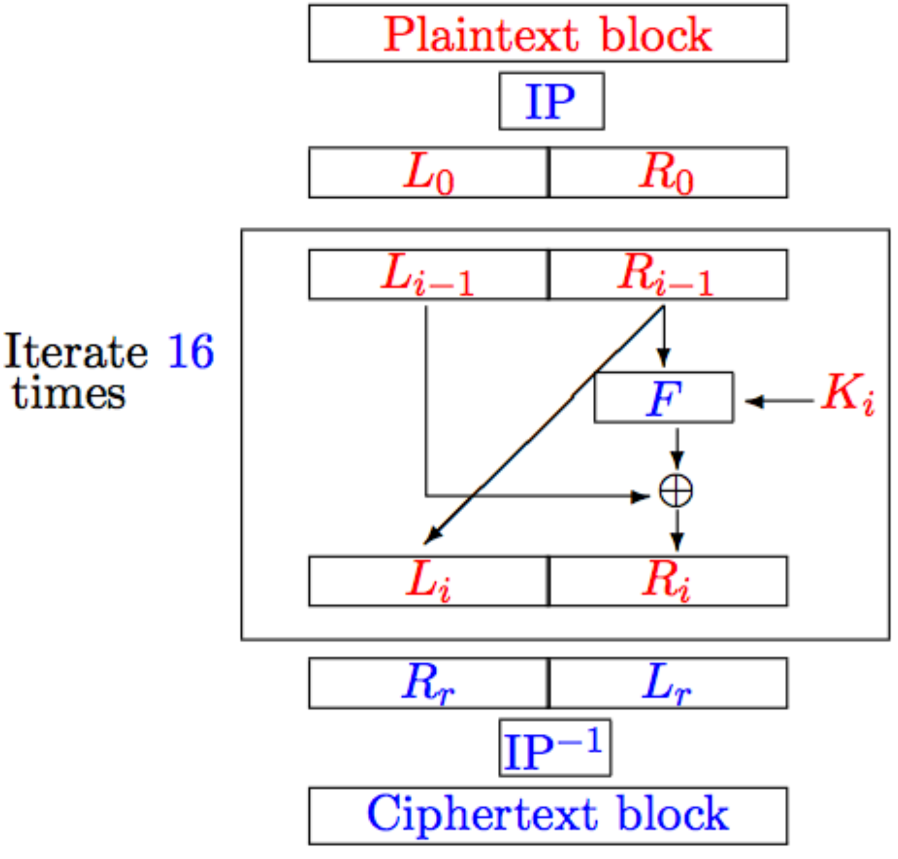
- Params: #rounds, Round key gen, Function F.
  - DES: 16 rounds, 64b block, 56b key, 48b round key



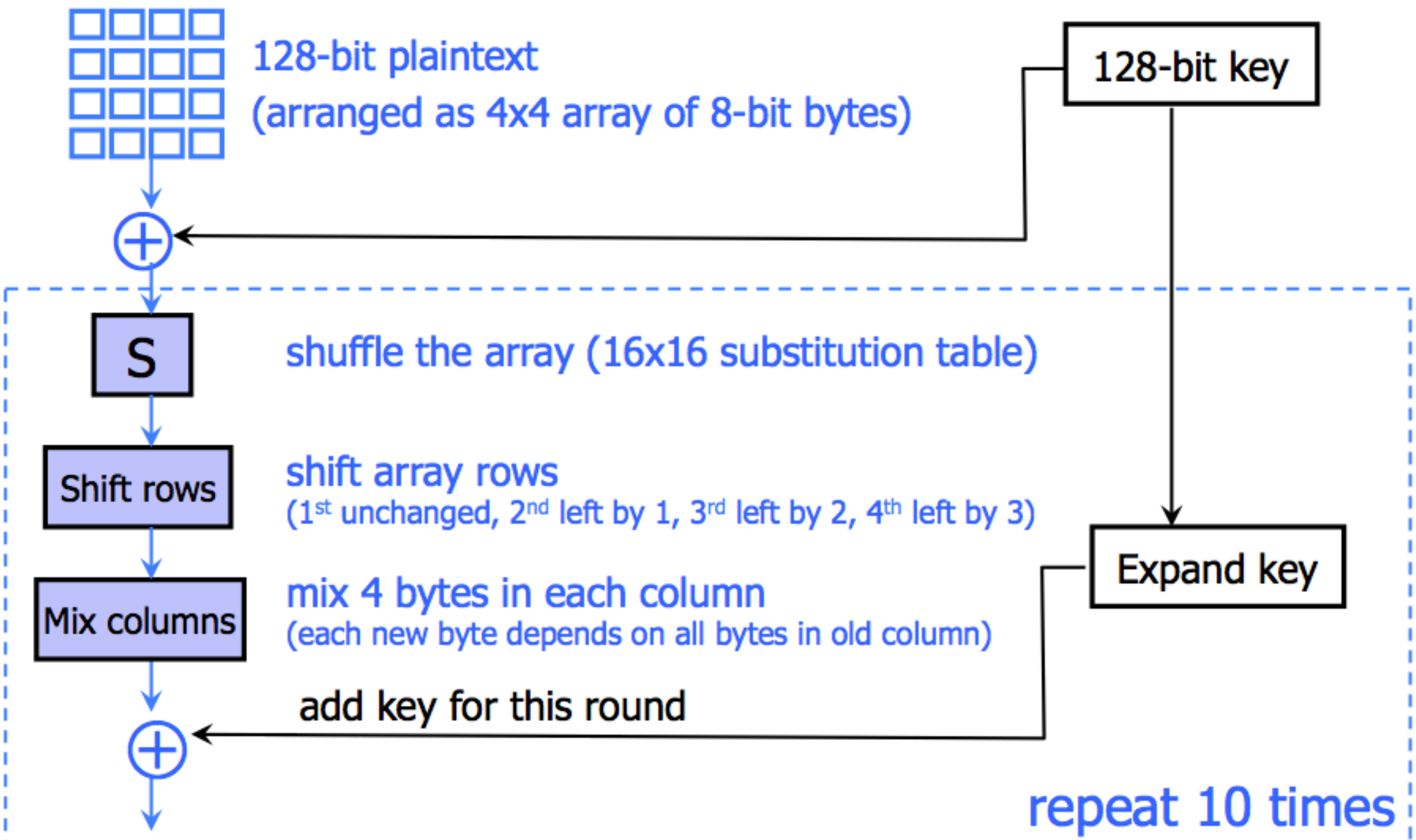
- Same code/circuit can be used for enc-dec, by reversing the order of Round-keys

DES

- Initial Permutation
- Split into L and R
- 16 rounds
- Join half blocks
- Final Permutation
- Function F:
  - Expansion,
  - Round key addition
  - Split + Sub. Box
  - Permute Box

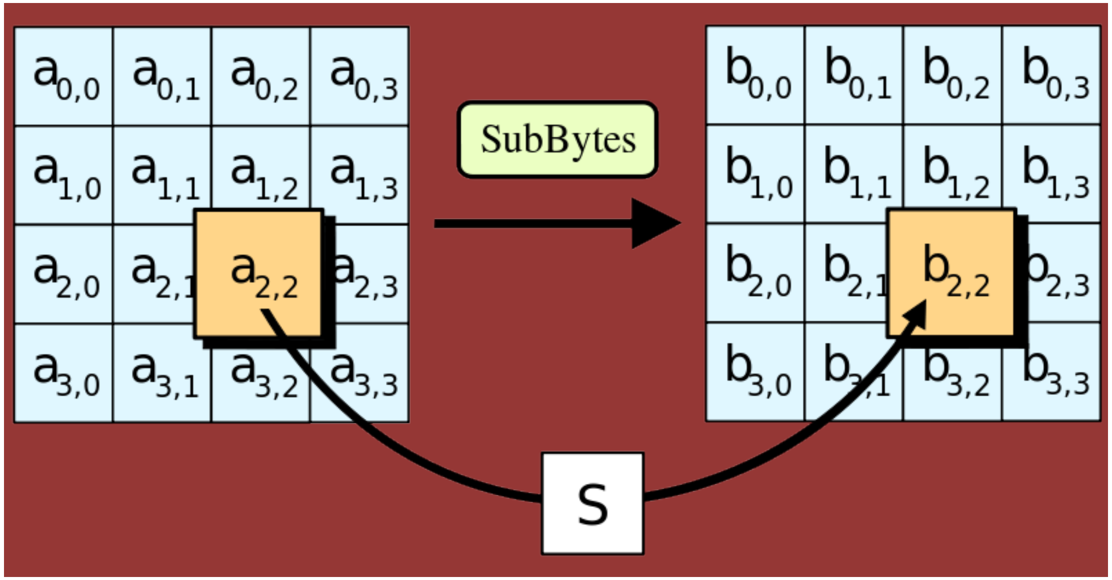


Rijndael/AES



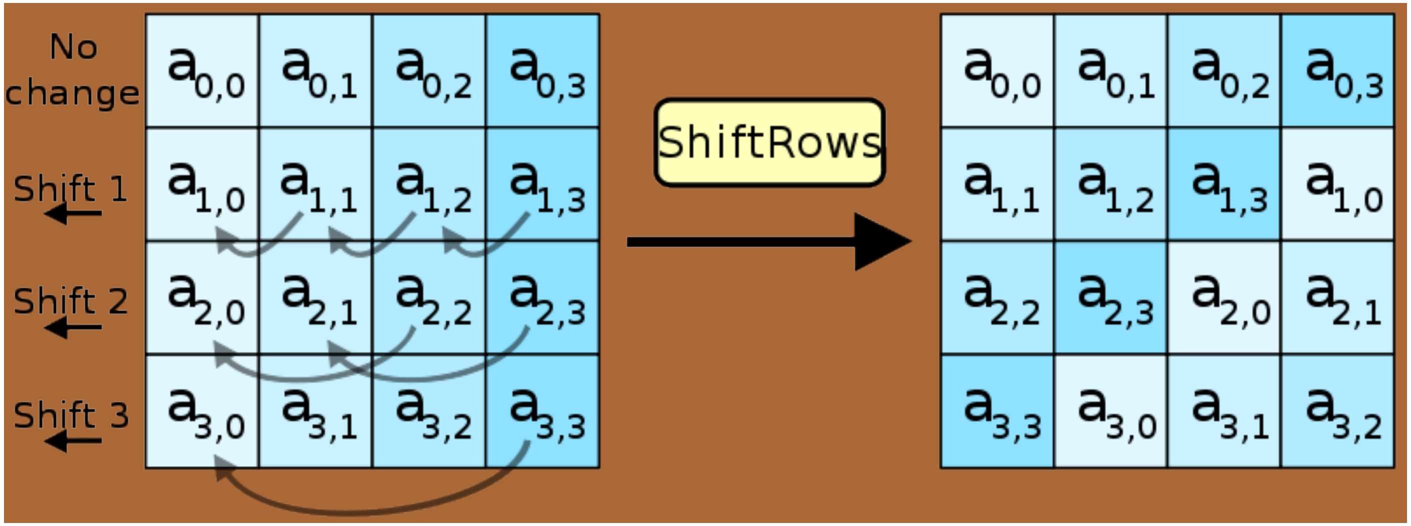
AES Steps: Substitution

- each byte in the *state* matrix is replaced with a SubByte using an 8-bit substitution box
- $b_{ij} = S(a_{ij})$



AES Steps: Shift Rows

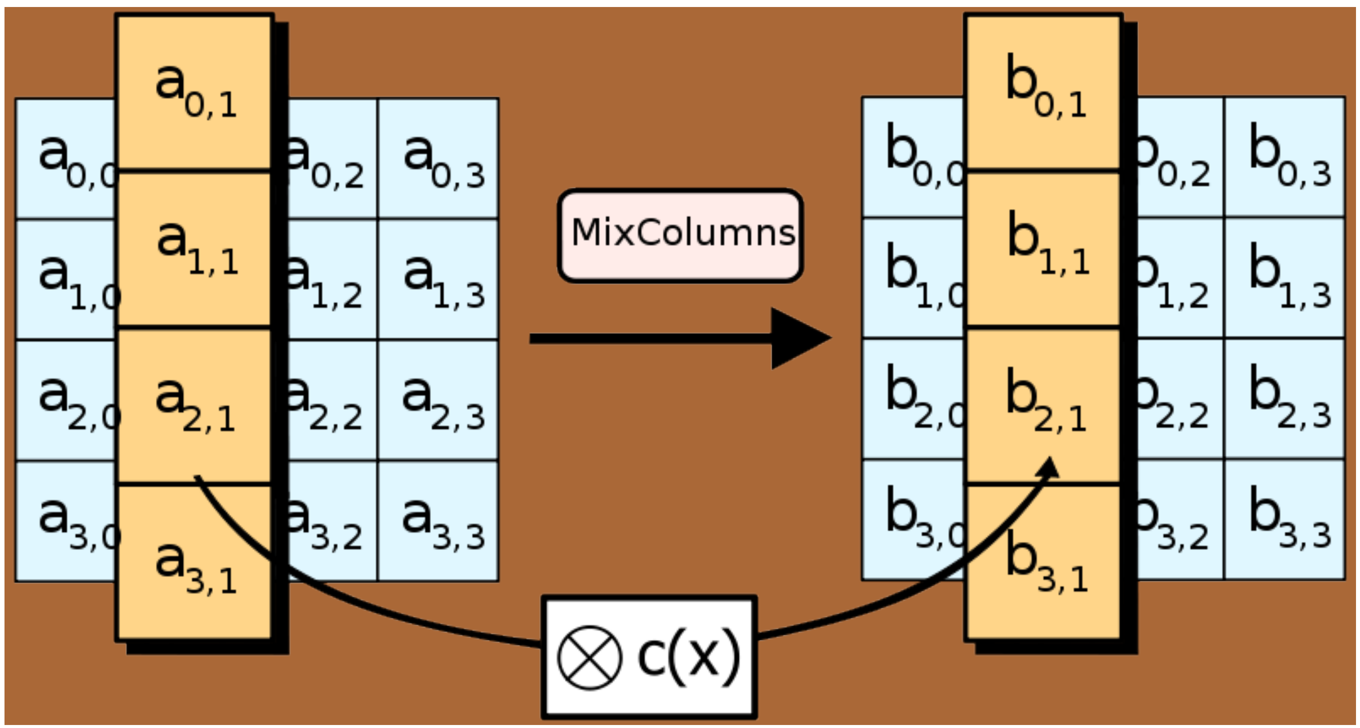
- Cyclically shifts the bytes in each row by a certain offset
- The number of places each byte is shifted differs for each row



AES Steps: Mix Columns

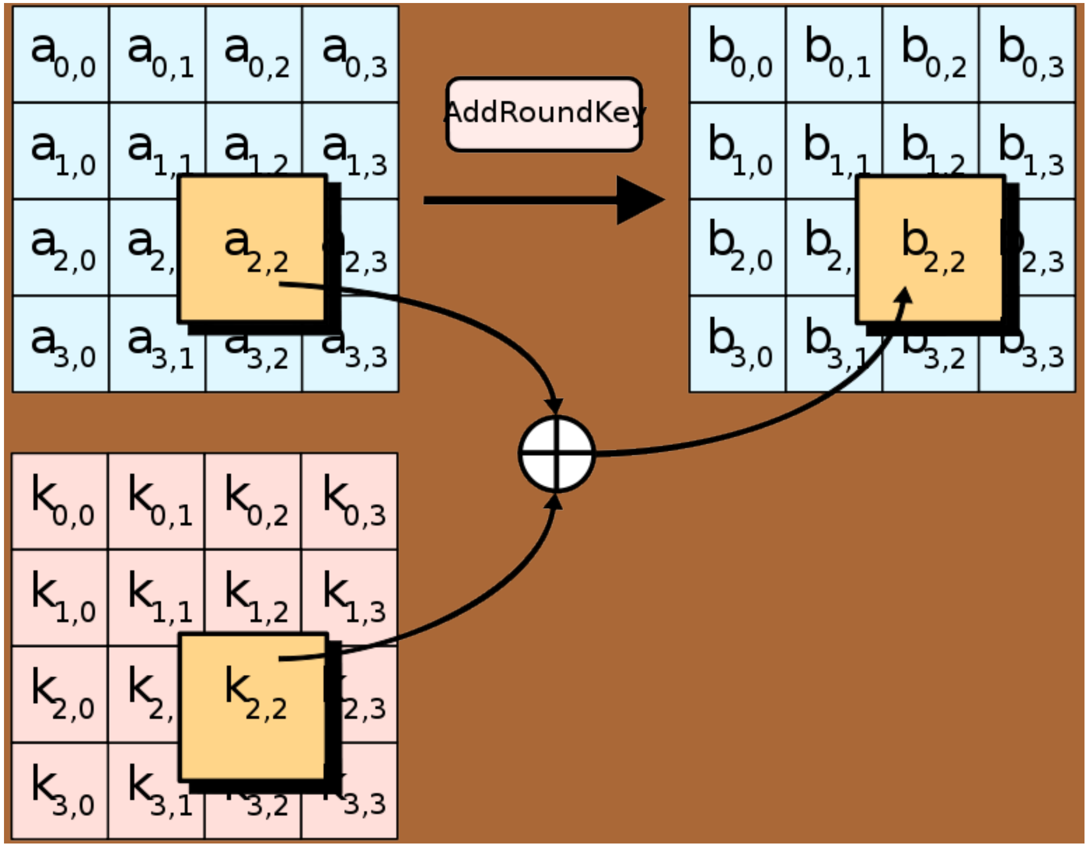
Each column is multiplied by the known matrix.
For the 128-bit key it is

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}.$$



AES Steps: Add Round Key

Each byte of the state is combined with a byte of the round subkey using the XOR operation



AES Security

- Brute Force Attack

Key size	Time to Crack
56-bit	399 seconds
128-bit	1.02×10^{18} years
192-bit	1.872×10^{37} years
256-bit	3.31×10^{56} years

- More common: side-channel attacks
  - Cache, power, EM, thermal, remanence …
  - S-box accesses, value of key bits, …

Fundamentals behind AES

- Prime field/ Galois field
  - Additive group w/ neutral element 0
  - Multiplicative group with neutral element 1
  - Distributive law a(b+c) = ab + ac
  - n = 1 in theorem below
  - Effectively: arithmetic modulo p

Theorem 4.3.1 *A field with order m only exists if m is a prime power, i.e., $m = p^n$, for some positive integer n and prime integer p. p is called the* characteristic *of the finite field.*

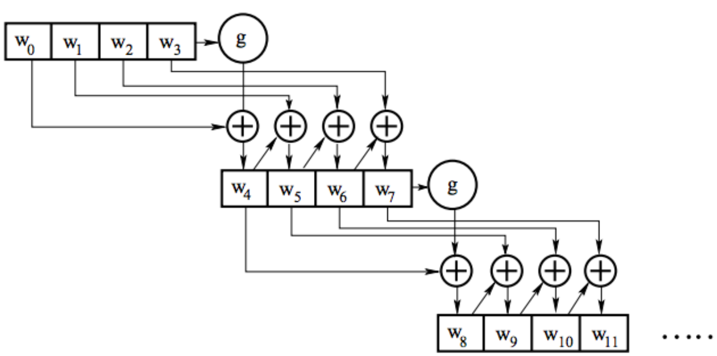
Understanding Cryptography (online textbook)

Polynomial Arithmetic

- m = 8 implies ‘extension fields’
- Each Byte is a polynomial with GF(2) coeff.
- Addition/Subtraction = XOR
- **Multiplication**: C(x) = A(x).B(x) mod P(x).
  - **Mix Columns.**
- P(x): **irreducible polynomial.** “prime”
  - $x^8+x^4+x^3+x+1$. x^4+x+1. *Not* $x^4 + x^3 + x + 1$.
- **GF(2^8) Inversion**: $A^{-1}(x)$. A(x) = 1 mod P(x)
  - **Substitution Box** (precomputed lookup tables)
  - only non-linear element in AES. S(a) + S(b) != S(a+b)

AES Layers

- **4x4 Bytes state.** 16B plaintext and round-keys
- **Substitution layer** S-box
  - one-one mapping (reqd for decryption)
  - A → GF(2^8) Inverse → Affine mapping → S(A)
- **Diffusion layer**: Shift Rows | Mix Columns
  - After 3 rounds, 16B plaintext → every byte of state
- **Key Addition layer**: xor with round key



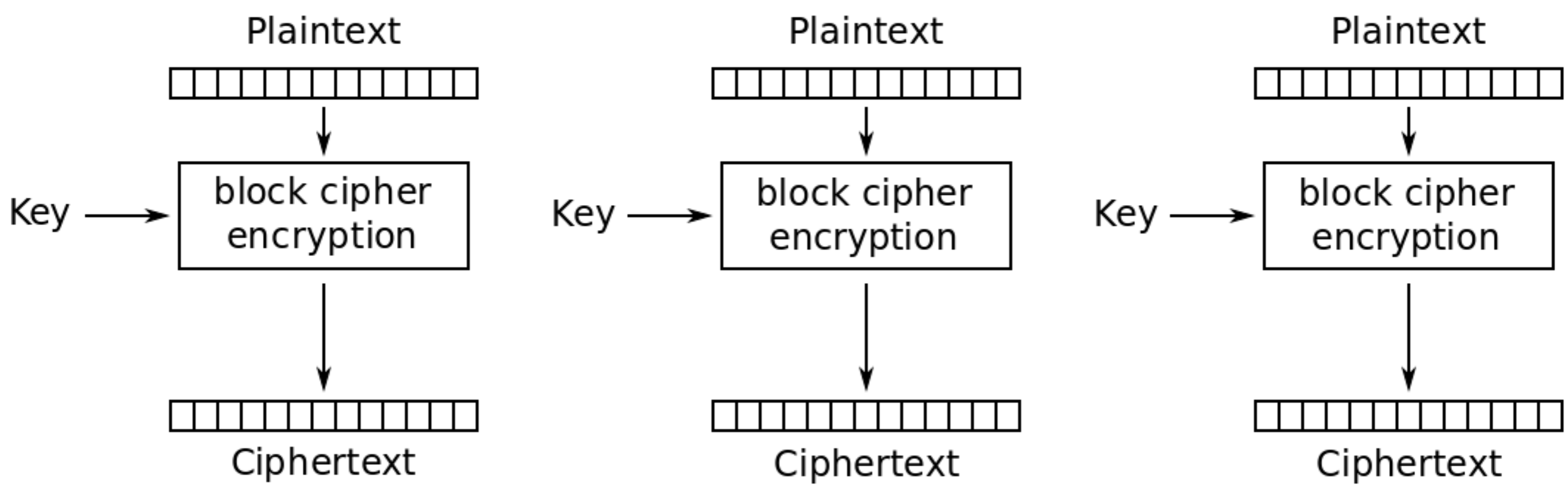
g(): * one-byte left circular rotation of word
* S-box
* xor with round-constant RC[i].

$$RC[1] = 0x01$$
$$RC[j] = 0x02 \times RC[j-1]$$

Encrypting a Large Message

- So, we've got a good block cipher, but our plaintext is larger than 128-bit block size
- Electronic Code Book (ECB) mode
  – Split plaintext into blocks, encrypt each one separately using the block cipher
- Cipher Block Chaining (CBC) mode
  – Split plaintext into blocks, XOR each block with the result of encrypting previous blocks
- Also various counter modes, feedback modes, etc.

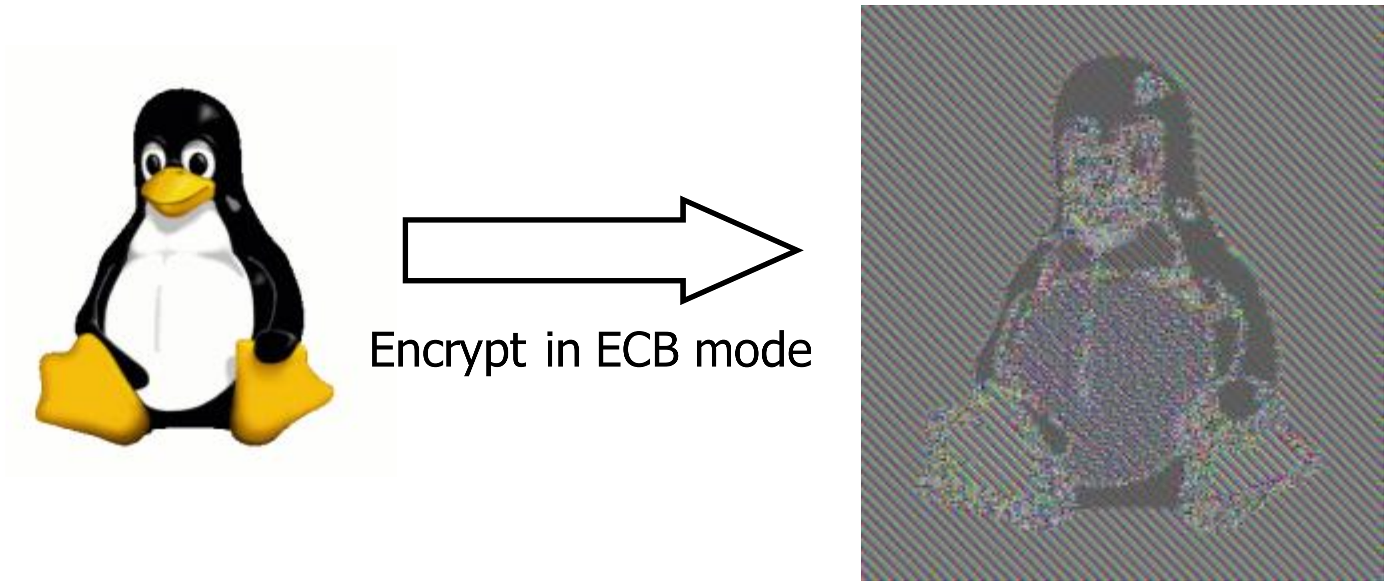
ECB Mode



- Identical blocks of plaintext produce identical blocks of ciphertext
- No integrity checks: can mix and match blocks

Information Leakage in ECB Mode

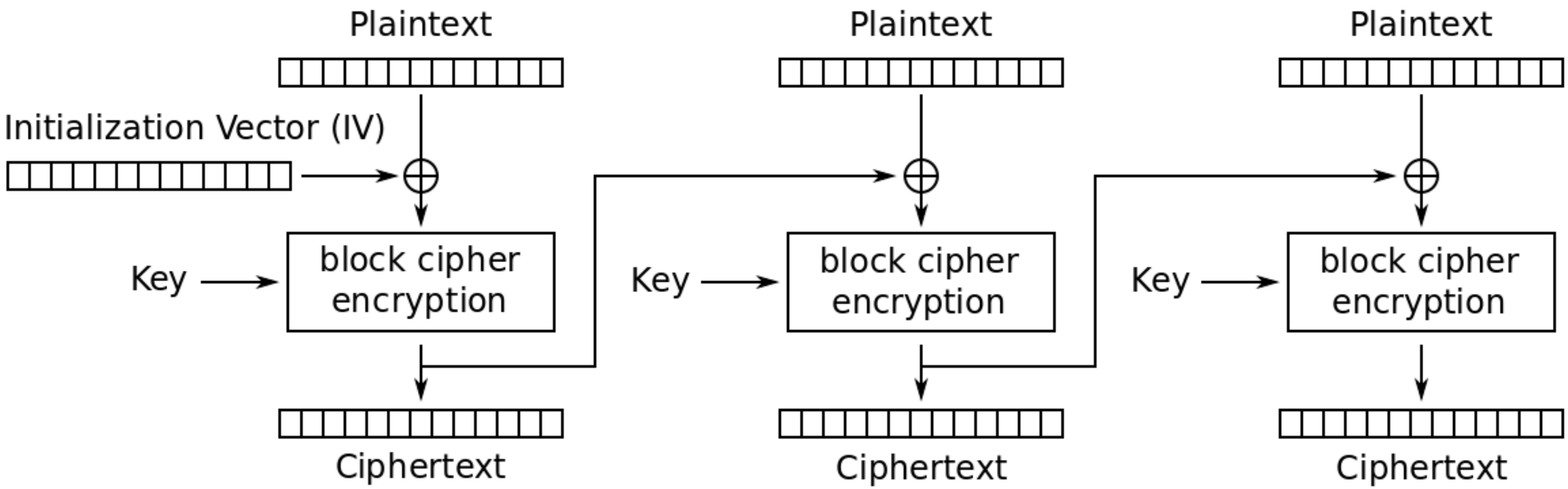
[Wikipedia]



Adobe Passwords Stolen (2013)

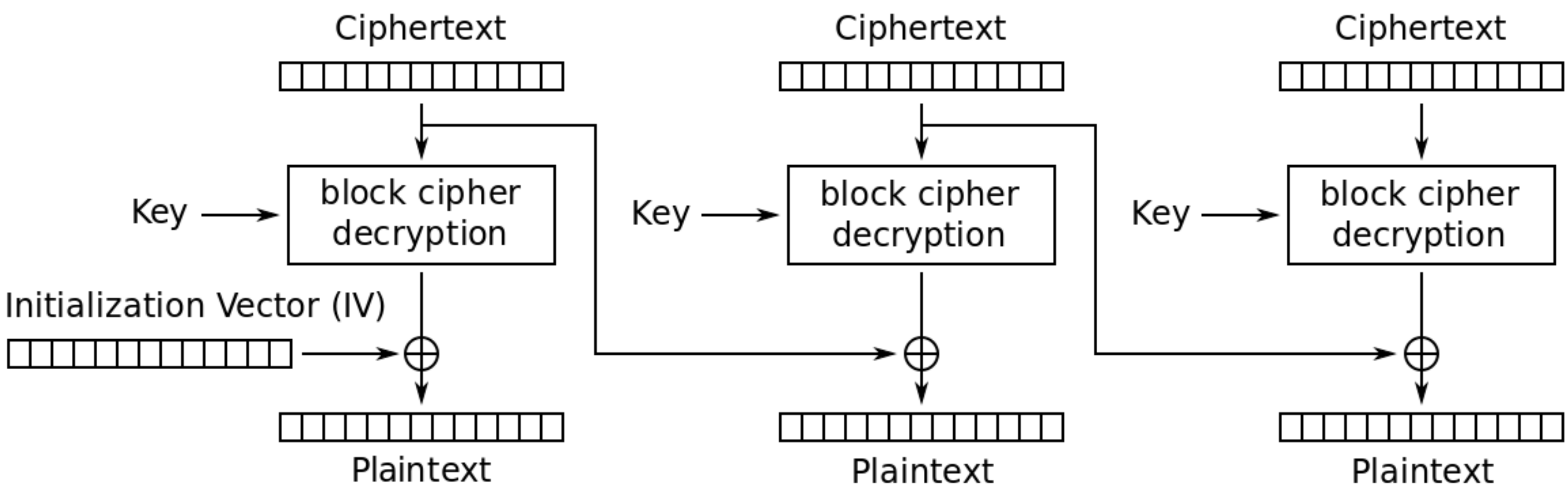
- 153 million account passwords
  - 56 million of them unique
- Encrypted using 3DES in ECB mode rather than hashed

CBC Mode: Encryption



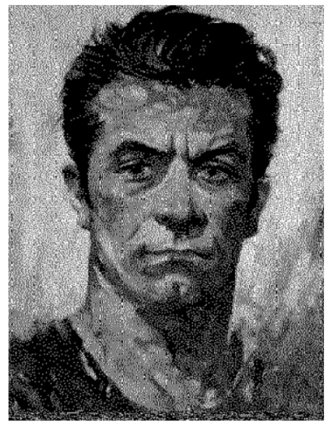
- Identical blocks of plaintext encrypted differently
- Last cipherblock depends on entire plaintext
  - Does not guarantee integrity

CBC Mode: Decryption



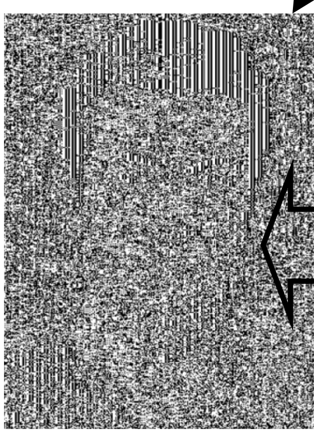
ECB vs. CBC

[Picture due to Bart Preneel]

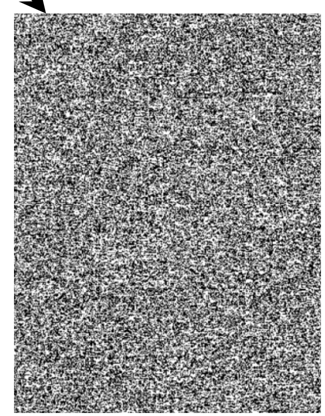


AES in ECB mode

AES in CBC mode



Similar plaintext blocks produce similar ciphertext blocks (not good!)



Choosing the Initialization Vector

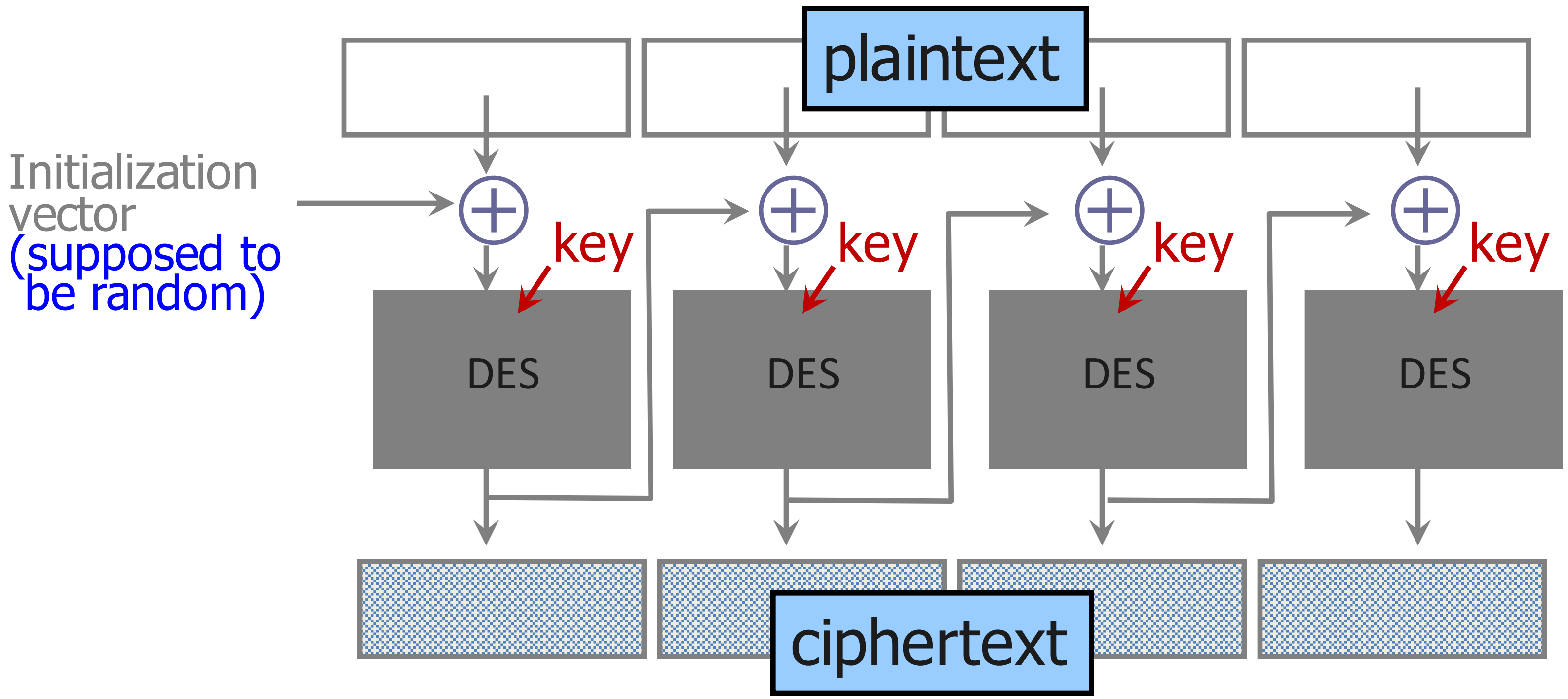
- Key used only once
  - No IV needed (can use IV=0)

- Key used multiple times
  - Best: fresh, random IV for every message
  - Can also use unique IV (eg, counter), but then the first step in CBC mode <u>must</u> be IV' $\leftarrow$ E(k, IV)
    - Example: Windows BitLocker
    - May not need to transmit IV with the ciphertext

- Multi-use key, unique messages
  - Synthetic IV: IV $\leftarrow$ F(k', message)
    - F is a cryptographically secure keyed pseudorandom function

CBC and Electronic Voting

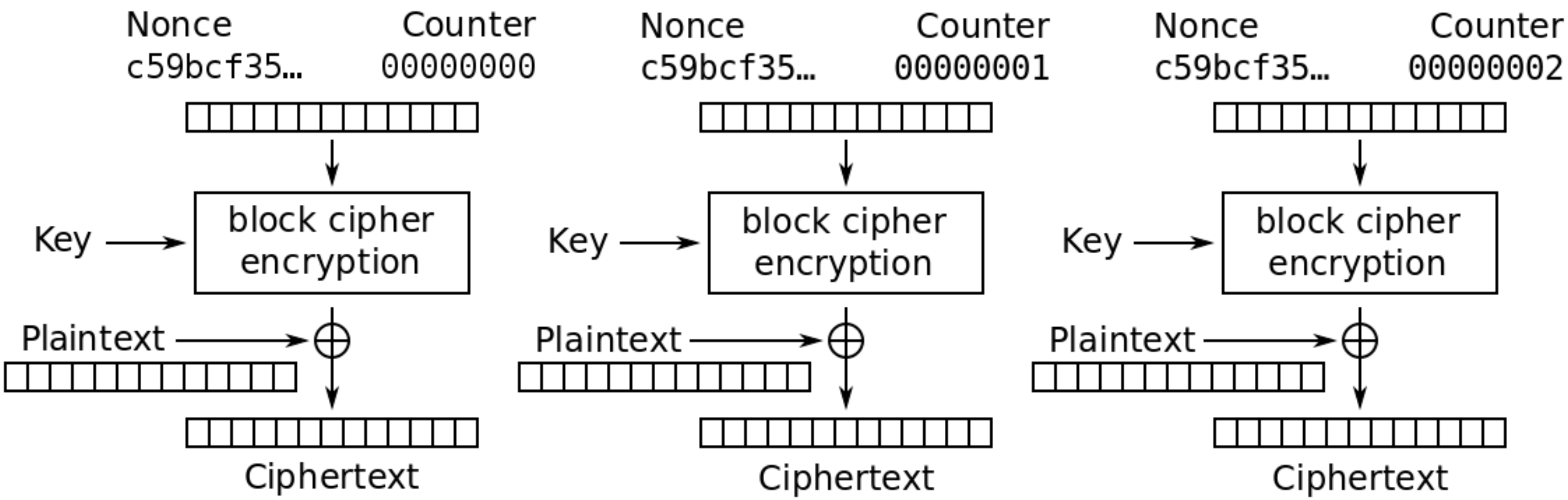
[Kohno, Stubblefield, Rubin, Wallach]



Found in the source code for Diebold voting machines:

```
DesCBCEncrypt((des_c_block*)tmp, (des_c_block*)record.m_Data,
        totalSize, DESKEY, NULL, DES_ENCRYPT)
```

CTR (Counter Mode)



- Does not guarantee integrity
- Fragile if counter repeats

How Can a Cipher Be Attacked?

- Attackers knows ciphertext and encryption algorithm
  - What else does the attacker know? Depends on the application in which the cipher is used!

- Known-plaintext attack (stronger)
  - Knows some plaintext-ciphertext pairs
- Chosen-plaintext attack (even stronger)
  - Can obtain ciphertext for any plaintext of his choice
- Chosen-ciphertext attack (very strong)
  - Can decrypt any ciphertext <u>except</u> the target
  - Sometimes very realistic

OpenSSL
Cryptography and SSL/TLS Toolkit

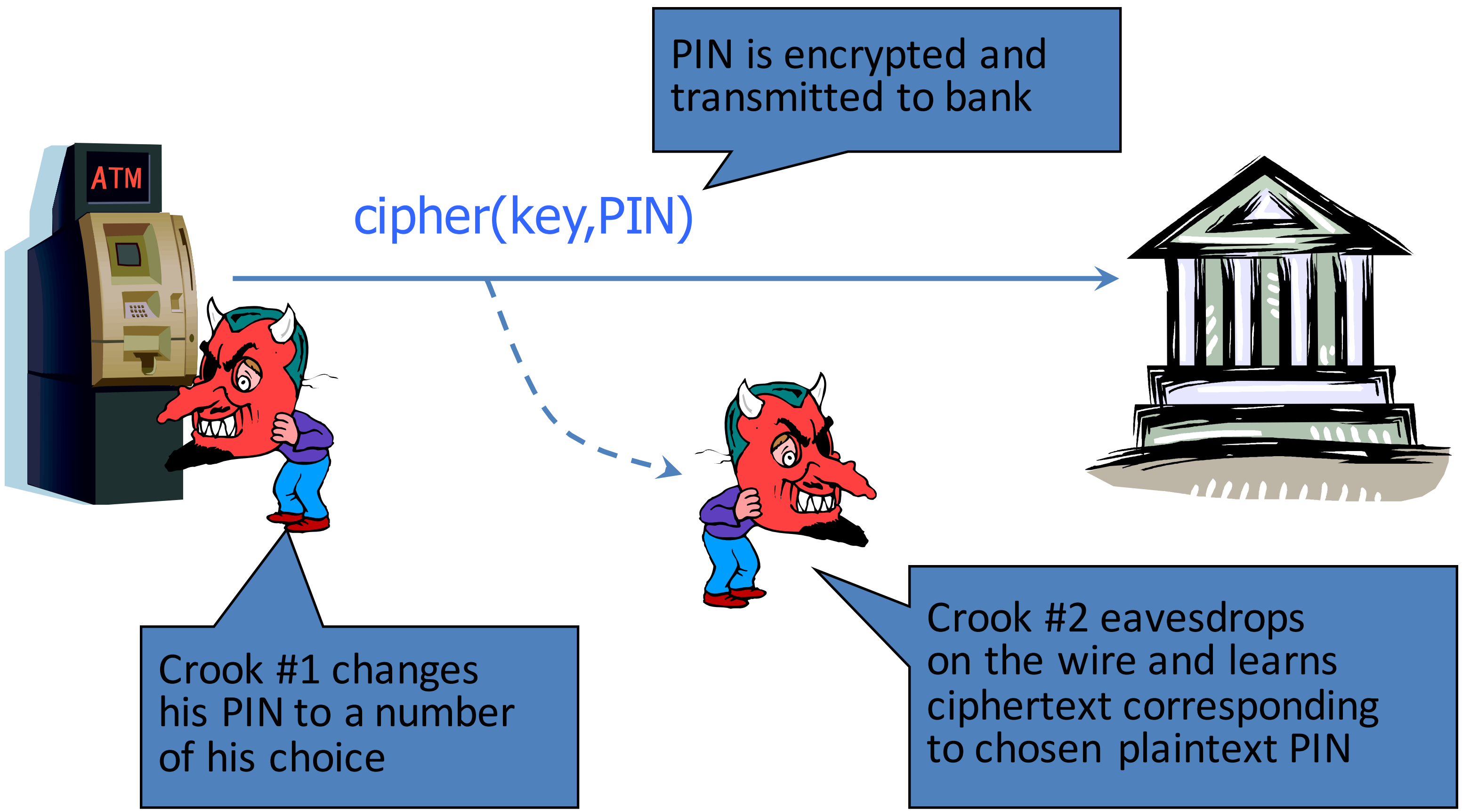
Known-Plaintext Attack

[From "The Art of Intrusion"]

Extracting password from an encrypted PKZIP file …

- "… I opened the ZIP file and found a `logo.tif' file, so I went to their main Web site and looked at all the files named `logo.tif.' I downloaded them and zipped them all up and found one that matched the same checksum as the one in the protected ZIP file"
- With known plaintext, PkCrack took 5 minutes to extract the key
  - Biham-Kocher attack on PKZIP stream cipher

Chosen-Plaintext Attack



… repeat for any PIN value

Security of Encryption Algos

- **Any deterministic, stateless symmetric encryption scheme is insecure**
  - Attacker can easily distinguish encryptions of different plaintexts from encryptions of identical plaintexts
  - This includes ECB mode of common block ciphers!

Attacker A interacts with Enc(-,-,b)

Let X,Y be any two different plaintexts

$C_1 \leftarrow$ Enc(X,X,b); $C_2 \leftarrow$ Enc(X,Y,b);

If $C_1=C_2$ then b=0 else b=1

- The advantage of this attacker A is 1

Prob(A outputs 1 if b=0)=0 Prob(A outputs 1 if b=1)=1

Key Distribution: Needham Schroeder

- Alice,Bob, trusted Server S, Nonce: random number used once.

$$
\begin{aligned}
A &\longrightarrow S : A, B, N_a, \\
S &\longrightarrow A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}, \\
A &\longrightarrow B : \{K_{ab}, A\}_{K_{bs}}, \\
B &\longrightarrow A : \{N_b\}_{K_{ab}}, \\
A &\longrightarrow B : \{N_b - 1\}_{K_{ab}}.
\end{aligned}
$$

- If adversary knows old session key, can replay session.

Key Distribution

- Authentication of one entity to another, and issue session keys
  - Separate auth from access control decisions
- Add timestamps.

$$
\begin{aligned}
A &\longrightarrow S : A, B, \\
S &\longrightarrow A : \{T_S, L, K_{ab}, B, \{T_S, L, K_{ab}, A\}_{K_{bs}}\}_{K_{as}}, \\
A &\longrightarrow B : \{T_S, L, K_{ab}, A\}_{K_{bs}}, \{A, T_A\}_{K_{ab}}, \\
B &\longrightarrow A : \{T_A + 1\}_{K_{ab}}.
\end{aligned}
$$

- Protocol verification: CSP, BAN logic etc.

Hash Functions

- Arbitrary length input → fixed length output
  - Integrity, Digital signature
- **Keyed hash**: message authentication code (MAC)
- **Requirements**
  - **Preimage resistant**: hard to find message with a given hash value
  - **Collision resistant**: hard to find two messages with the same hash value
  - **Second preimage resistant**: Given one message, hard to find another with the same hash value.

Merkle-Damgard Construction

- **Iterate over blocks (similar to CBC mode).**

$l = s - n$

Pad the input message m with zeros so that it is a multiple of l bits in length

Divide the input m into t blocks of l bits long, $m_1, \ldots, m_t$

Set H to be some fixed bit string of length n.

for $i = 1$ **to** t **do**

  $H = f(H \| m_i)$

end

return (H)

- **Length strengthening**: Pad zero bits to create N blocks, then a final block of L bits to encode the original length of unpadded message

SHA-1

- Not recommended anymore.

Google Security Blog

Announcing the first SHA1 collision

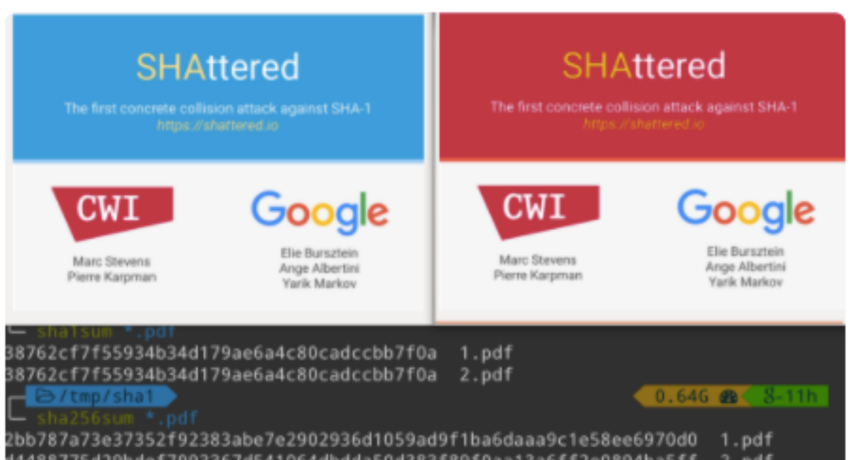
February 23, 2017

http://shattered.io/

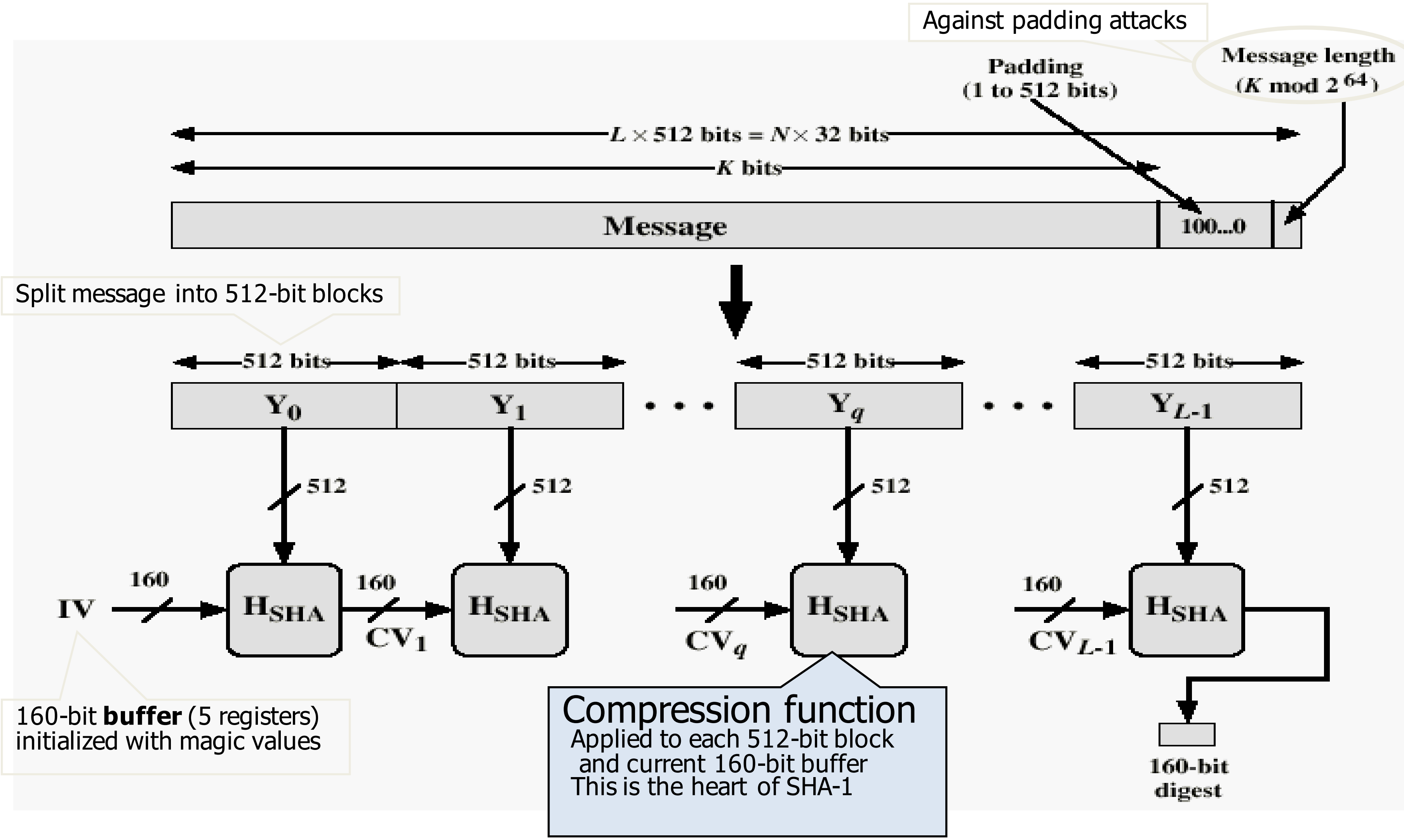
MD5
1 smartphone
30 sec

SHA-1 Shattered
110 GPU
1 year

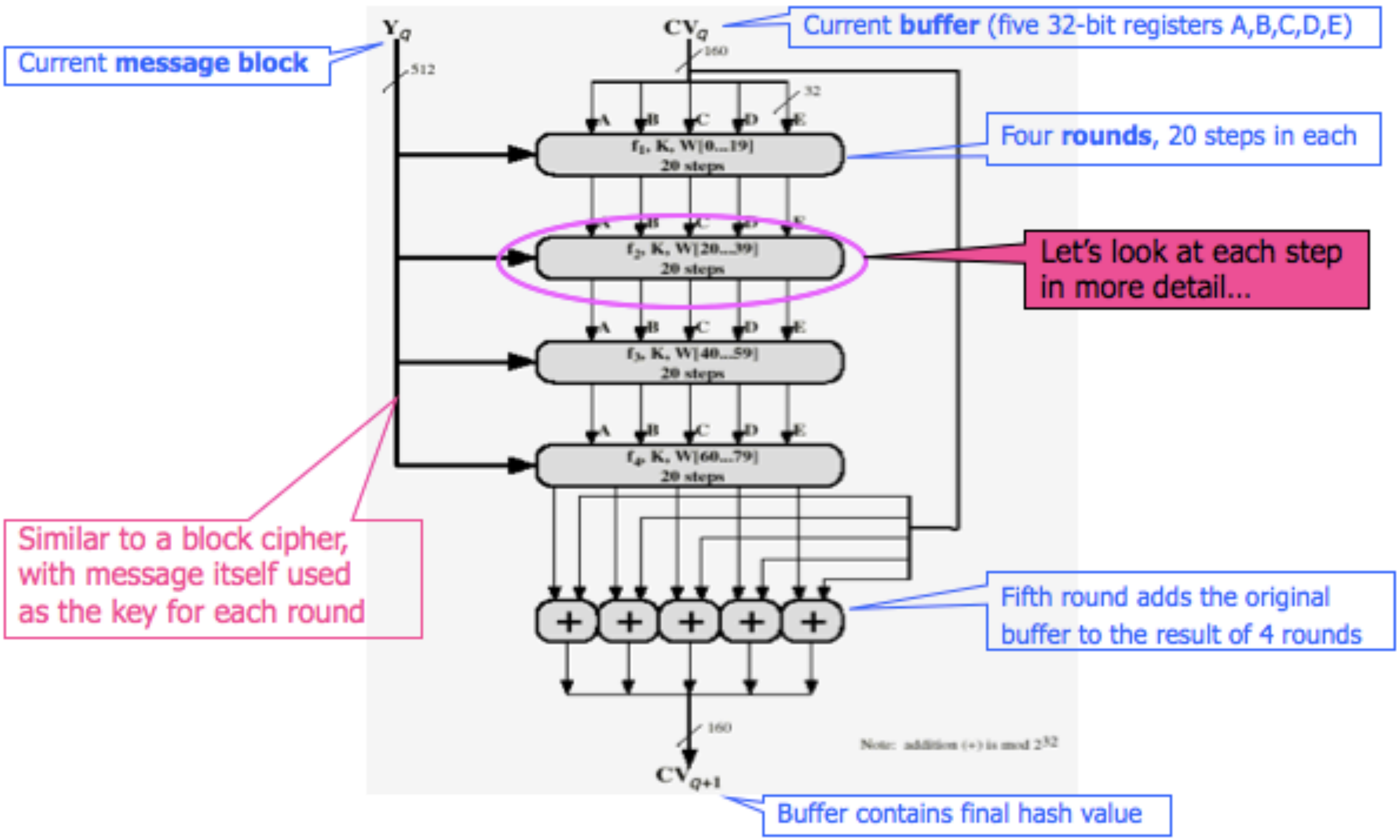
SHA-1 Bruteforce
12.000.000 GPU
1 year



SHA-1 [shmatikov]

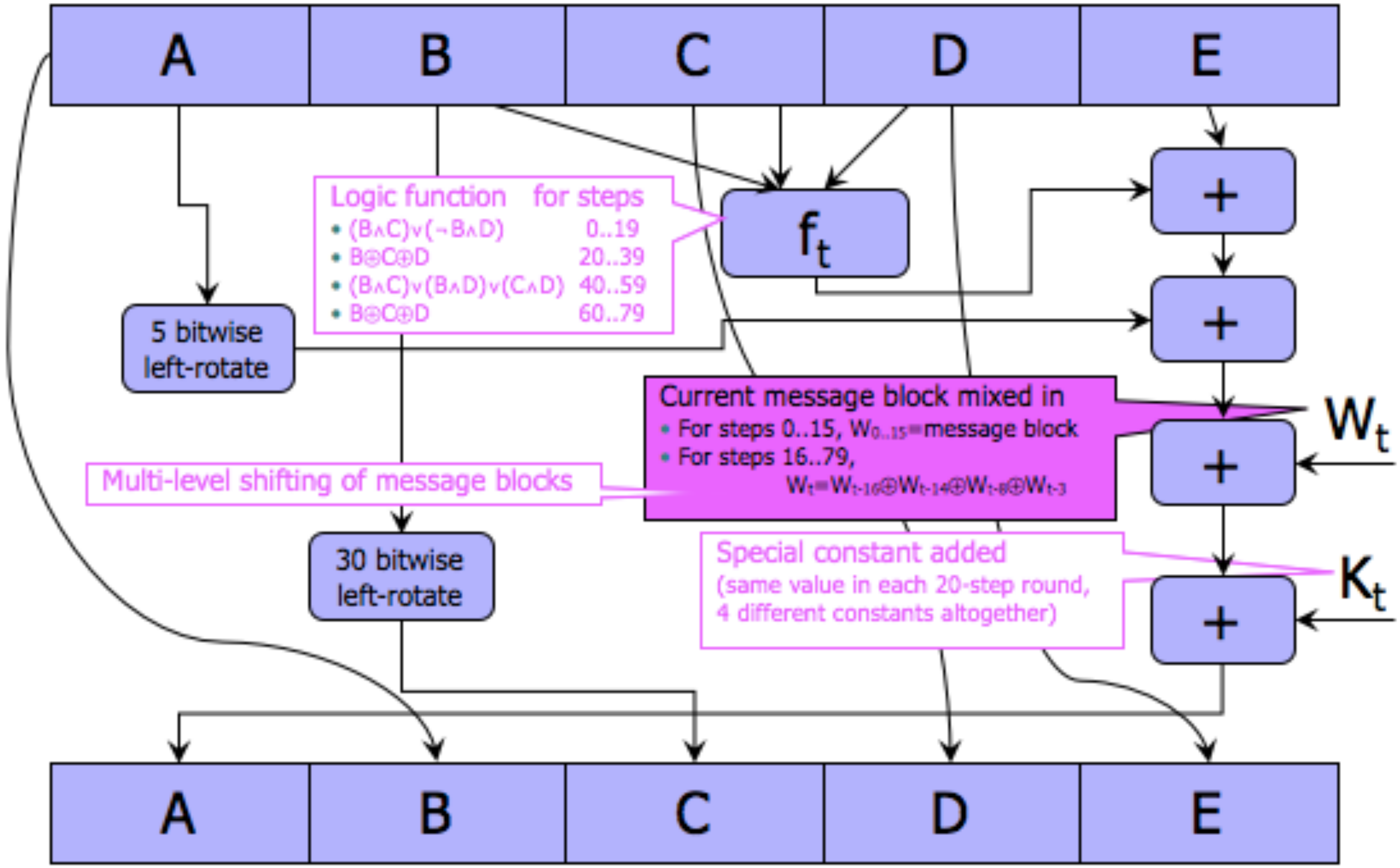


SHA-1



Current message block
Current buffer (five 32-bit registers A,B,C,D,E)
Four rounds, 20 steps in each
Let's look at each step in more detail…
Similar to a block cipher, with message itself used as the key for each round
Fifth round adds the original buffer to the result of 4 rounds
Buffer contains final hash value

Each Step of SHA-1 (of 80 steps)



SHA-1

- Not recommended anymore.

$(A, B, C, D, E) = (H_1, H_2, H_3, H_4, H_5)$
/* Expansion */
for $j = 16$ **to** 79 **do**
$\quad X_j = ((X_{j-3} \oplus X_{j-8} \oplus X_{j-14} \oplus X_{j-16}) \lll 1)$
end
Execute Round 1
Execute Round 2
Execute Round 3
Execute Round 4
$(H_1, H_2, H_3, H_4, H_5) = (H_1 + A, H_2 + B, H_3 + C, H_4 + D, H_5 + E)$

SHA-1 Round functions

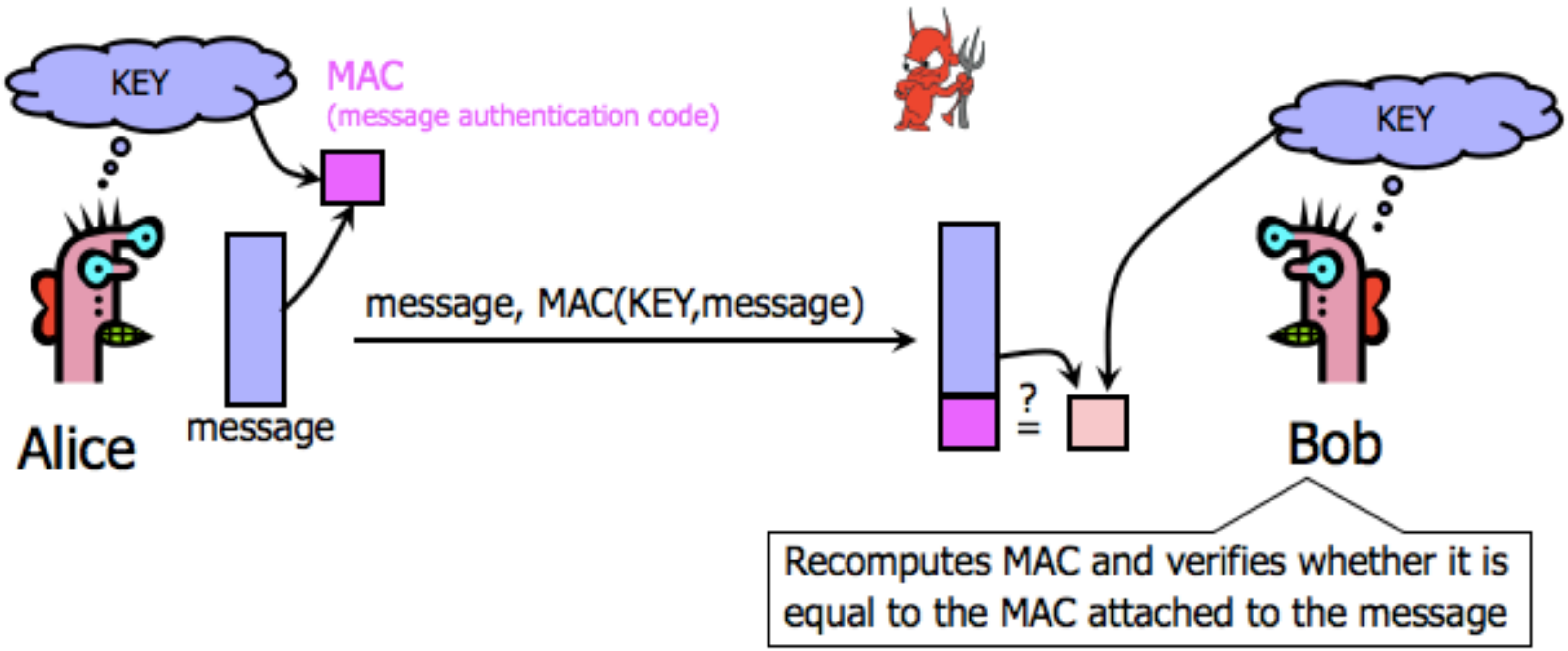
Round 1
for $j = 0$ **to** 19 **do**
$t = (A \lll 5) + f(B, C, D) + E + X_j + y_1$
$(A, B, C, D, E) = (t, A, B \lll 30, C, D)$
end
Round 2
for $j = 20$ **to** 39 **do**
$t = (A \lll 5) + h(B, C, D) + E + X_j + y_2$
$(A, B, C, D, E) = (t, A, B \lll 30, C, D)$
end
Round 3
for $j = 40$ **to** 59 **do**
$t = (A \lll 5) + g(B, C, D) + E + X_j + y_3$
$(A, B, C, D, E) = (t, A, B \lll 30, C, D)$
end
Round 4
for $j = 60$ **to** 79 **do**
$t = (A \lll 5) + h(B, C, D) + E + X_j + y_4$
$(A, B, C, D, E) = (t, A, B \lll 30, C, D)$
end

Hash Function Family

- Differ in rounds and constants, but similar structure.

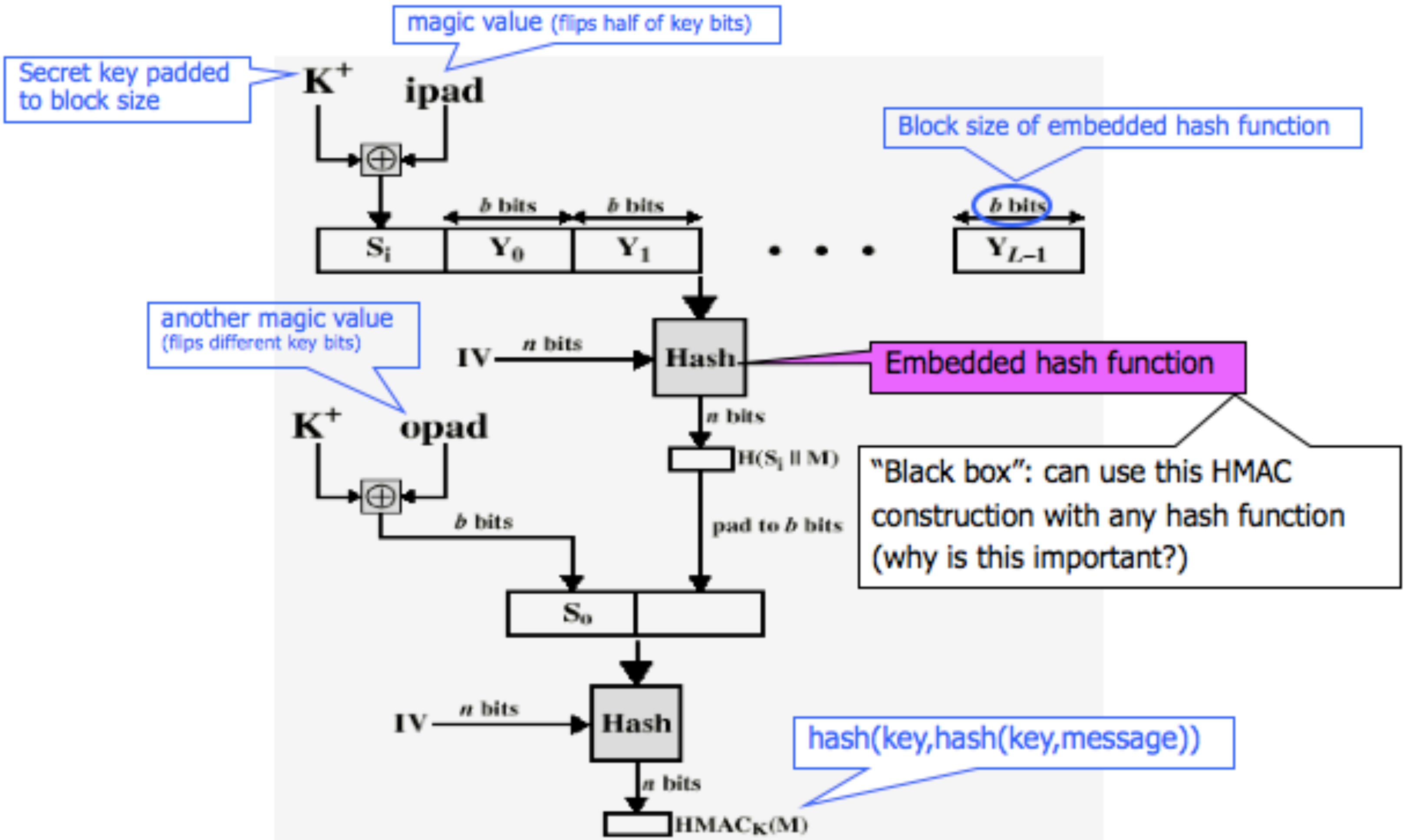
- **MD4**: This has 3 rounds of 16 steps and an output bitlength of 128 bits.
- **MD5**: This has 4 rounds of 16 steps and an output bitlength of 128 bits.
- **SHA-1**: This has 4 rounds of 20 steps and an output bitlength of 160 bits.
- **RIPEMD-160**: This has 5 rounds of 16 steps and an output bitlength of 160 bits.
- **SHA-256**: This has 64 rounds of single steps and an output bitlength of 256 bits.
- **SHA-384**: This is identical to SHA-512 except the output is truncated to 384 bits.
- **SHA-512**: This has 80 rounds of single steps and an output bitlength of 512 bits.

MACs with Authentication



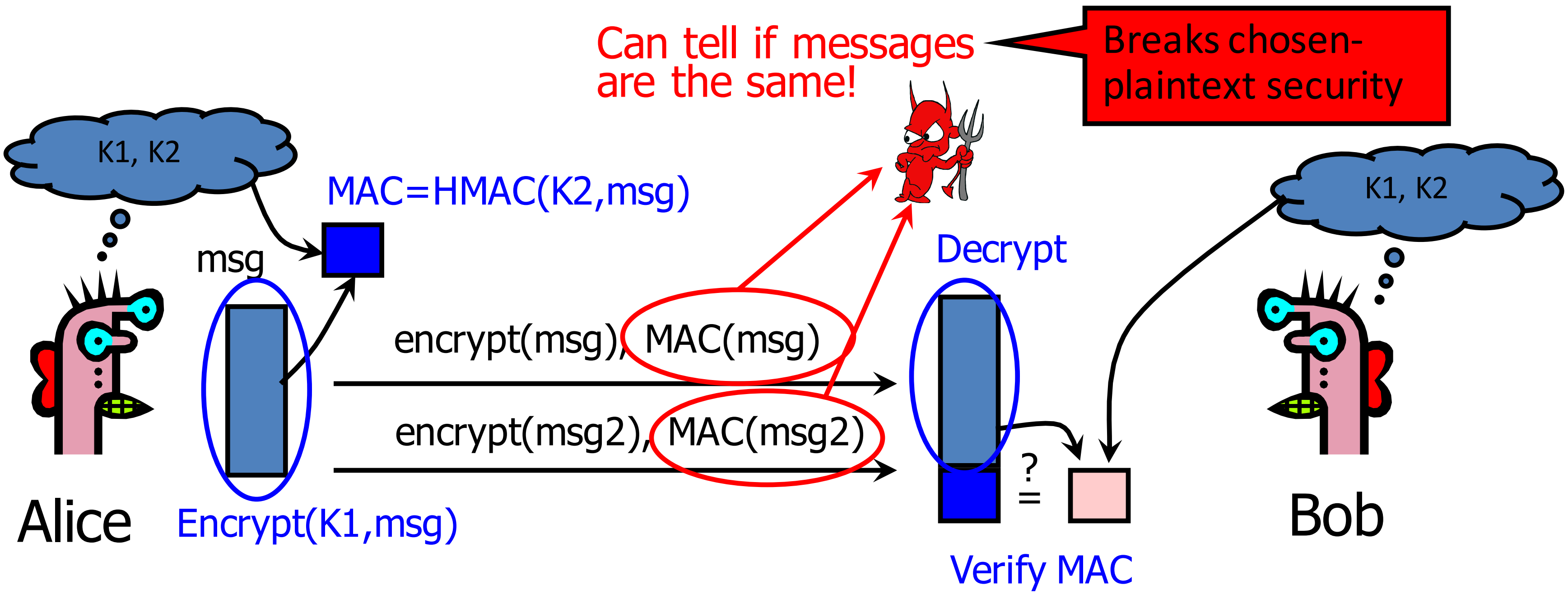
Integrity and authentication: only someone who knows KEY can compute correct MAC for a given message

Keyed-MAC (HMAC)



Encrypt + MAC

Goal: confidentiality + integrity + authentication



MAC is deterministic: messages are equal ⇒ their MACs are equal

Solution: Encrypt, <u>then</u> MAC (or MAC, then encrypt)

Asymmetric Crypto

- Encryption for confidentiality
  - Private:public key pair
- Digital signatures for authentication and integrity
  - Alice signs using private key, Bob verifies using public key
- Key management and Certificate Authorities
  - Session keys: e.g. Diffie-Hellman key exchange.

Diffie Hellman (Merkle)

Diffie–Hellman Set-up

1. Choose a large prime p.
2. Choose an integer $\alpha \in \{2, 3, \ldots, p-2\}$.
3. Publish p and α.

Diffie–Hellman Key Exchange

Alice		**Bob**
choose $a = k_{pr,A} \in \{2, \ldots, p-2\}$		choose $b = k_{pr,B} \in \{2, \ldots, p-2\}$
compute $A = k_{pub,A} \equiv \alpha^a \bmod p$		compute $B = k_{pub,B} \equiv \alpha^b \bmod p$
	$\xrightarrow{k_{pub,A}=A}$	
	$\xleftarrow{k_{pub,B}=B}$	
$k_{AB} = k_{pub,B}^{k_{pr,A}} \equiv B^a \bmod p$		$k_{AB} = k_{pub,A}^{k_{pr,B}} \equiv A^b \bmod p$

RSA: Rivest Shamir Adleman

- 1977. Independently created in '73
- **Key generation**
  - Two primes p, q
  - n = p.q, $\varphi(n) = \varphi(p)\varphi(q) = (p - 1)(q - 1)$
  - e, coprime with d, s.t. $d \cdot e \equiv 1 \pmod{\varphi(n)}$
  - Public key (n,e). Private key (n, d)
- **Encryption of m: $c = m^e \bmod n$**
- **Decryption of c: $c^d \bmod n = (m^e)^d \bmod n = m$**

RSA Decryption [shmatikov]

$e \cdot d \equiv 1 \bmod \varphi(n)$

Thus $e \cdot d = 1+k \cdot \varphi(n) = 1+k(p-1)(q-1)$ for some k

If gcd(m,p)=1, then by Fermat's Little Theorem, $m^{p-1} \equiv 1 \bmod p$

Raise both sides to the power k(q-1) and multiply by m, obtaining $m^{1+k(p-1)(q-1)} \equiv m \bmod p$

Thus $m^{ed} \equiv m \bmod p$

By the same argument, $m^{ed} \equiv m \bmod q$

Since p and q are distinct primes and $p \cdot q = n$,

$m^{ed} \equiv m \bmod n$ (chinese remainder theorem)

RSA and Factoring

- Given n, factor into p & q, and hence $\varphi(n)$
- Hence, with e and $d \cdot e \equiv 1 \pmod{\varphi(n)}$, get d.
- Solution to factoring breaks RSA
  - But RSA problem is to recover m from c
  - Taking e^{th} root of c modulo n
  - Might break without factoring as well. Unknown.

‘Textbook’ RSA is Bad

- Deterministic
  - Attacker can guess plaintext, compute ciphertext, and compare for equality
  - If messages are from a small set (for example, yes/no), can build a table of corresponding ciphertexts
- Can tamper with encrypted messages
  - Take an encrypted auction bid c and submit $c(101/100)^e \bmod n$ instead
- Does not provide semantic security (security against chosen-plaintext attacks)

RSA + Integrity

- "Textbook" RSA does not provide integrity
  - Given encryptions of m_1 and m_2, attacker can create encryption of $m_1 \cdot m_2$
    - $(m_1^e) \cdot (m_2^e) \bmod n \equiv (m_1 \cdot m_2)^e \bmod n$
  - Attacker can convert m into m^k without decrypting
    - $(m^e)^k \bmod n \equiv (m^k)^e \bmod n$
- In practice, OAEP is used: instead of encrypting M, encrypt $M \oplus G(r) \,;\, r \oplus H(M \oplus G(r))$
  - r is random and fresh, G and H are hash functions
  - Resulting encryption is plaintext-aware: infeasible to compute a valid encryption without knowing plaintext
    - … if hash functions are "good" and RSA problem is hard

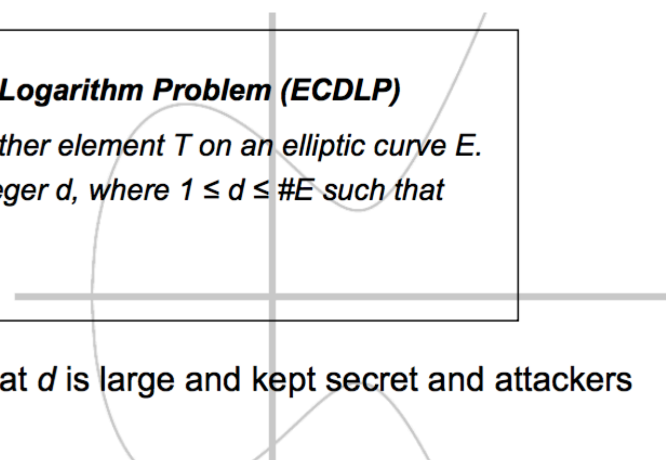
Other Trapdoor One-way Fns

- **Elliptic-curves: gen. of discrete log problem**
  - Shorter keys, faster than RSA-1024+
  - Points on an elliptic curve (+ extra pt at infinity) form cyclic sub-groups
  - To generate a curve with about 2^{160} points, a prime with a length of about 160 bits is required

Definition: Elliptic Curve Discrete Logarithm Problem (ECDLP)

Given a primitive element P and another element T on an elliptic curve E. The ECDL problem is finding the integer d, where 1 ≤ d ≤ #E such that

$$\underbrace{P + P + \ldots + P}_{d\ times} = dP = T.$$



- Cryptosystems are based on the idea that *d* is large and kept secret and attackers cannot compute it easily

Summary

- **Key exchange**
  - Protocols, certificate authority
- **Asymmetric**
  - Used for key exchange, can encrypt or sign
- **Symmetric**
  - Session encryption, can use to sign as well (not rec.)
- **Signatures/MACs/Digest** (keyed/otherwise)
  - Fast vs. Slow

Next: Memory Errors

- Input maliciously crafted values to target → take control over target's execution

- **Many sub-categories:**
  - Code injection
  - Control-flow
  - Data-flow

- **Baseline defenses**
  - Data execution prevention
  - Address-space randomization
  - Control-flow integrity
  - Memory safety