

Overview of Cryptography Concepts

Charlie Kaufman and Radia Perlman

radia@alum.mit.edu

charliekaufman@outlook.com

Radia.Perman@dell.com

What do we need cryptography for?

- Uses
 - Encryption
 - Integrity protection
 - Authentication

We'll talk about three building blocks

- Secret key cryptography
- Public key cryptography
- Hashes

Secret Key Cryptography

Secret Key Cryptography

- Two inverse functions, “encrypt” and “decrypt”
- A single key that both Alice (the encrypter) and Bob (the decrypter) must know
- Intuitively like using multiplication and division as the two functions... encrypt by multiplying by the key, decrypt by dividing by the key
- But multiplication and division as the algorithm wouldn't be secure, and it's much nicer if the numbers are integers
- So, the cryptographers have come up with secure and efficient algorithms, with numbers that can be represented in a fixed number of bits

All Cryptography is only as secure as the computational resources of the attacker

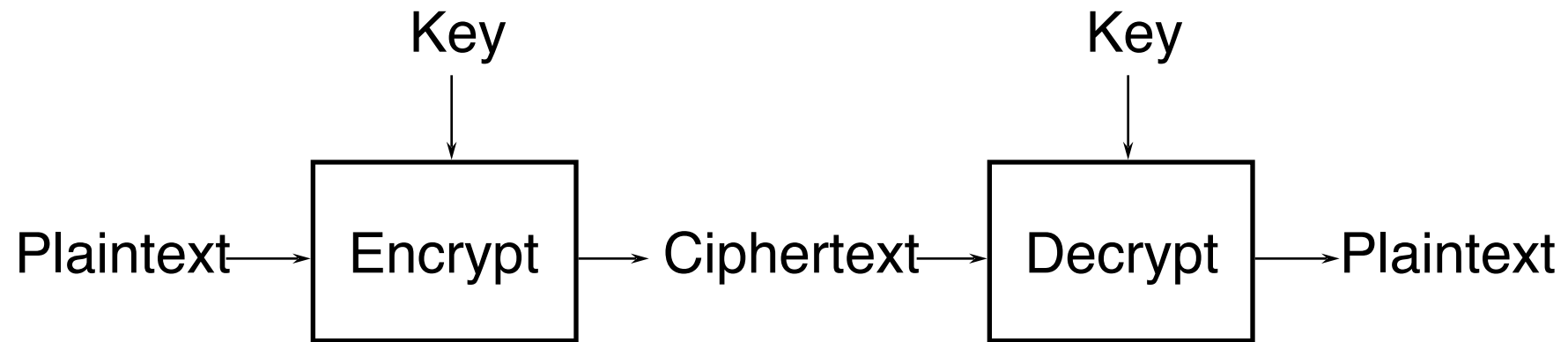
- If an attacker could go through all possible keys, he could find the one you're using and see your messages
- This is made infeasible by having enough different keys to overwhelm the capabilities of any attacker
- Surprisingly, keys don't have to be that big if they are randomly chosen
 - 128-bits is big enough for any conceivable attack today
- An encryption algorithm is considered secure only if there is no way to decrypt the message significantly more easily than trying all keys

What can you do with secret key cryptography?

- Encrypt and decrypt
- Authentication
- Integrity protection

Our notation for encryption

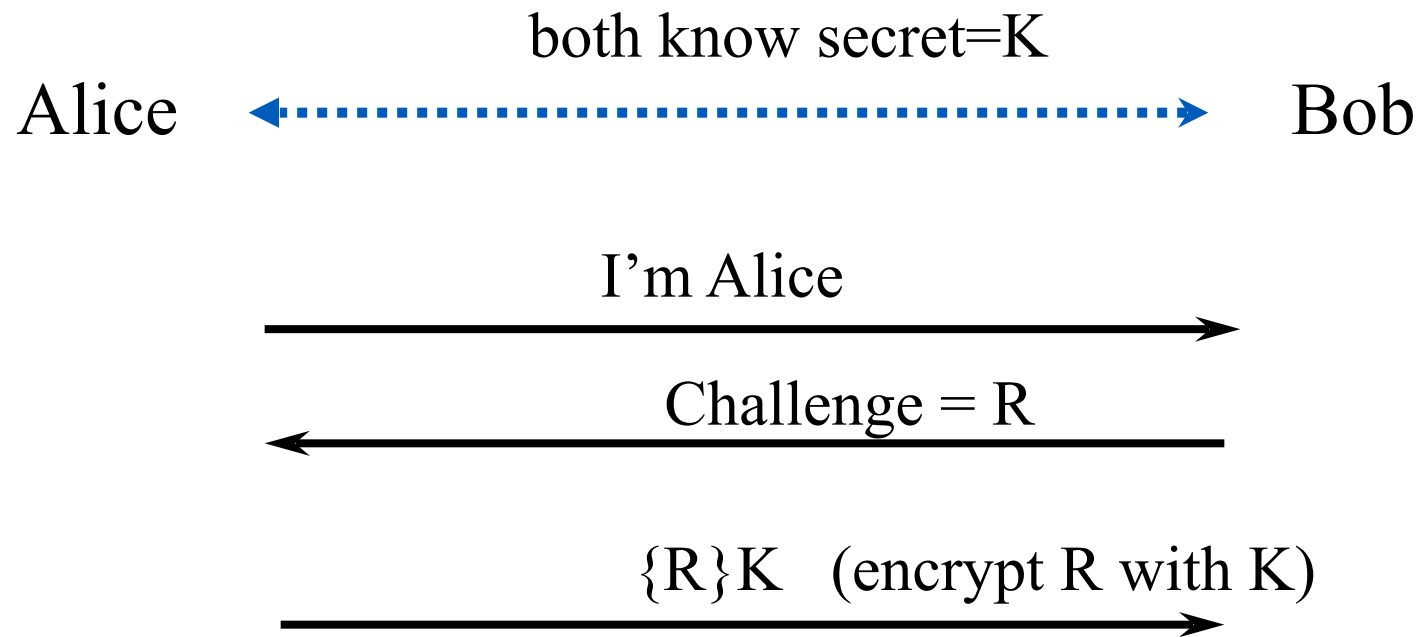
- Curly brackets used for encryption, with name of key following, so $\{msg\}K$ means “msg” encrypted with key K



Authentication with Password

- If you're talking to the wrong thing, they now know your password
- Also, you are divulging it to an eavesdropper
- A better way is to use cryptography

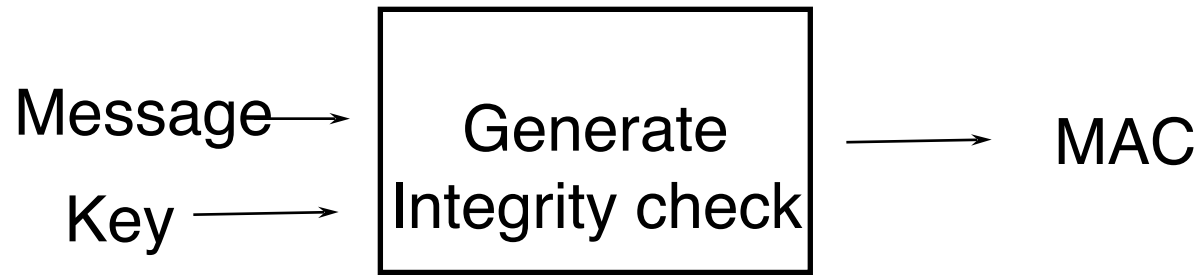
Cryptography allows you to prove you know the secret without saying the secret



Integrity protection

- Bob wants to know that the message sent by Alice hasn't been altered
- Again, Bob and Alice both know the secret key K
- The integrity check is a function of the message and the secret K
- It's like a checksum, except a checksum only detects random errors made by faulty hardware rather than changes carefully crafted by a clever attacker

Integrity Check (MAC---Message Authentication Code)



The most widely deployed secret key algorithm today

- AES (Advanced Encryption Standard)
- AES is actually 3 algorithms, one with 128-bit keys, one with 192-bit keys, and one with 256-bit keys
- There are others, e.g., DES was the standard for many years
- DES needed to be replaced

Public Key Cryptography

Public Key Cryptography

- Also known as “asymmetric cryptography”
- Secret key is like having two functions, inverses of each other
 - Similar to using multiplication and division; using the same secret
- Public key uses two keys which are inverses of each other
 - “public key”, and “private key”
 - Everyone knows Bob’s public key; only Bob knows his private key
 - Analogy; nobody invented division – only known function is multiplication
 - Bob’s public key is 5, his private key is $1/5$
 - To encrypt for Bob, multiply by his public key (5). He can undo it using his private key ($1/5$)
- Cryptographers have come up with clever algorithms that accomplish the goals of public key cryptography

Public key Uses

- Encryption
 - Encrypt for Bob using his public key
 - Bob undoes the encryption using his private key
- Authentication (Bob proves he knows his private key)
 - Alice can tell Bob to decrypt or sign a challenge
- Integrity check: (known as a digital signature if using public keys)
 - Bob generates a digital signature on “msg” using his private key
 - Alice verifies Bob’s digital signature using his public key (and “msg” and signature)
 - Our notation for message signed by Bob: [msg]Bob

Performance Issue

- Public key algorithms are much less efficient than secret key
- So, you wouldn't encrypt a large message with a public key
- Instead, Alice would encrypt a secret key S (a random number for the message she's sending) with Bob's public key, and send the large message encrypted with S
 - Alice sends: $\{S\}_{\text{Bob}}$ and $\{\text{msg}\}_S$

Some important public key algorithms today

- RSA
- Diffie-Hellman
- ECC (Elliptic Curve Cryptography)
- Note that all these would no longer be secure if there were a sufficiently large quantum computer
- So they are being replaced by new algorithms that accomplish the same goals (more later)

Hash Algorithms

What's a cryptographic hash?

- Input: a message of arbitrary size
- Output: a fixed size quantity, e.g., 256 bits
- It should have these properties
 - Efficient to compute
 - Given a value h , should be computationally infeasible to find a message that has hash value h
 - Why isn't it easy to find a message with a particular hash?
 - Even though zillions of arbitrary sized messages will have the same 256-bit hash value, with a good cryptographic hash, the only way to find one is brute force search
 - “Collision resistant”: should be infeasible to find two messages with the same hash

Uses of Hashes

Making digital signatures practical

- Digital signatures are done using public key cryptography
- Remember, public key cryptography is very slow compared to hashes and secret key
- Computing a digital signature (using public key cryptography) on a big message would be impractical
- The hash of the message will be a small fixed size (e.g., 256 bits or 512 bits)
- Bob digitally signs the hash of the message; not the message itself
- Which means that if someone could find a different message with the same hash as something Bob signed, then Bob has signed the other message too!

Another hash use: Integrity check

- Assume Alice and Bob share a secret K . Alice can use $\text{hash}(\text{msg}, K)$ as the integrity check on the message
- Only someone that knows K can generate or verify the integrity check

Current state of Hash Functions

- Hash functions turn out to be useful in lots of other cryptographic contexts
- There are two standard ones (SHA-2 and SHA-3), each with variants of different output sizes ranging from 256 to 512 bits
- Lots of others still deployed

Certificates and PKI

PKI: Public Key Infrastructure

- The stuff that lets Bob reliably learn Alice's public key
- It's not a secret... Alice could publish it!
- ...but someone could also publish some different key and claim it is Alice's. How does Bob tell the difference?
- Bob needs a trusted way of knowing Alice's public key

How does Bob know Alice's public key?

- A certificate is a small message signed by something Bob trusts, that says "Alice's public key is 2374928710"
- The message might include other information, like an expiration date
- The thing that signs certificates is known as a CA (Certificate Authority)
- Bob is configured with the public key of the CA (or perhaps of many CAs)
- Alice, having chosen her (public, private) key pair, gets a CA to sign a certificate mapping her name to her public key
- Bob sends a message to Alice saying he wants to talk
- Alice sends her certificate to Bob
- Bob (being configured with the CA public key), verifies the signature on the certificate, and now trusts that Alice's public key is 2374928710 (the value in the certificate)

How can this be used to establish a session?

- In a very simple scheme (used in SSL), Bob randomly chooses a secret K , and sends it to Alice, encrypted with her public key
- The remainder of the conversation is then cryptographically protected with K

Design of key distribution without public keys

Why would you want to do without public keys?

- Patents were a problem before 2000
- Performance remains an issue in some cases
- Kerberos was a design that avoided using public key cryptography
- A “KDC” (Key Distribution Center) was configured with secrets for each principal
- Each principal just needed to know the one secret key it shares with the KDC

KDC's database

KDC

Alice/ K_A
Bob/ K_B
Carol/ K_C

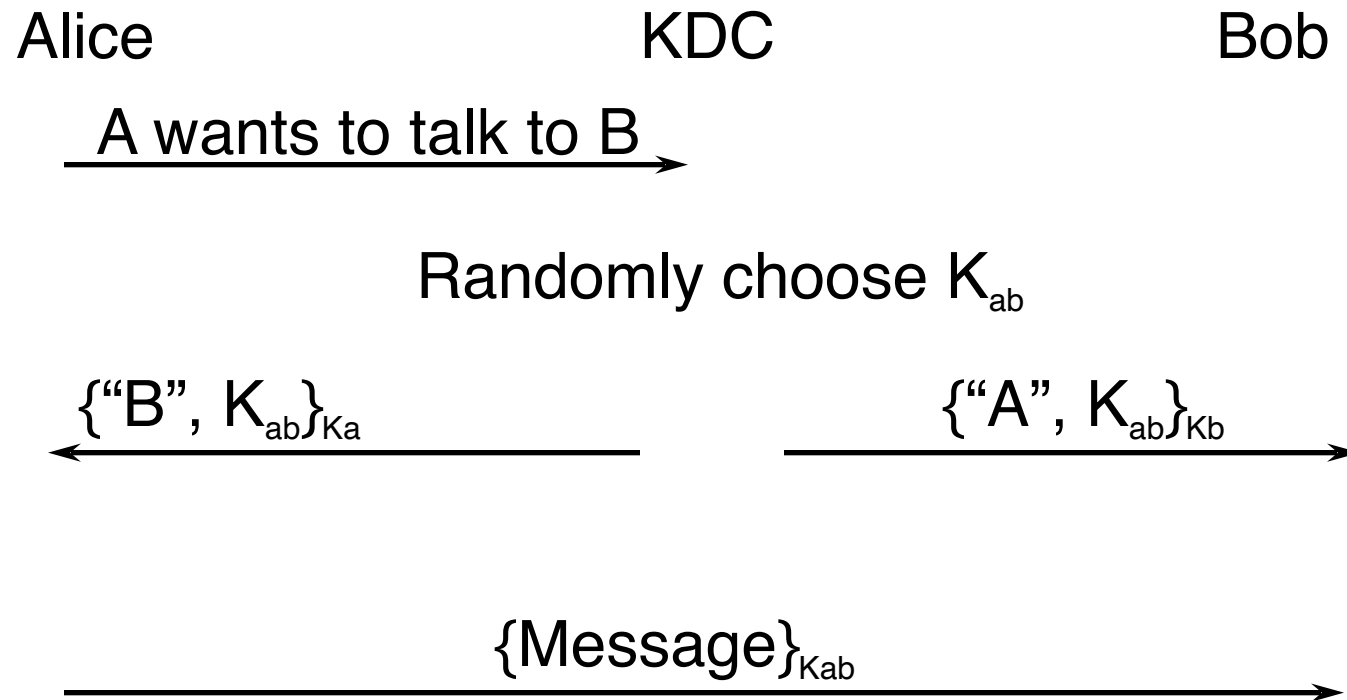
Alice
 K_A

Bob
 K_B

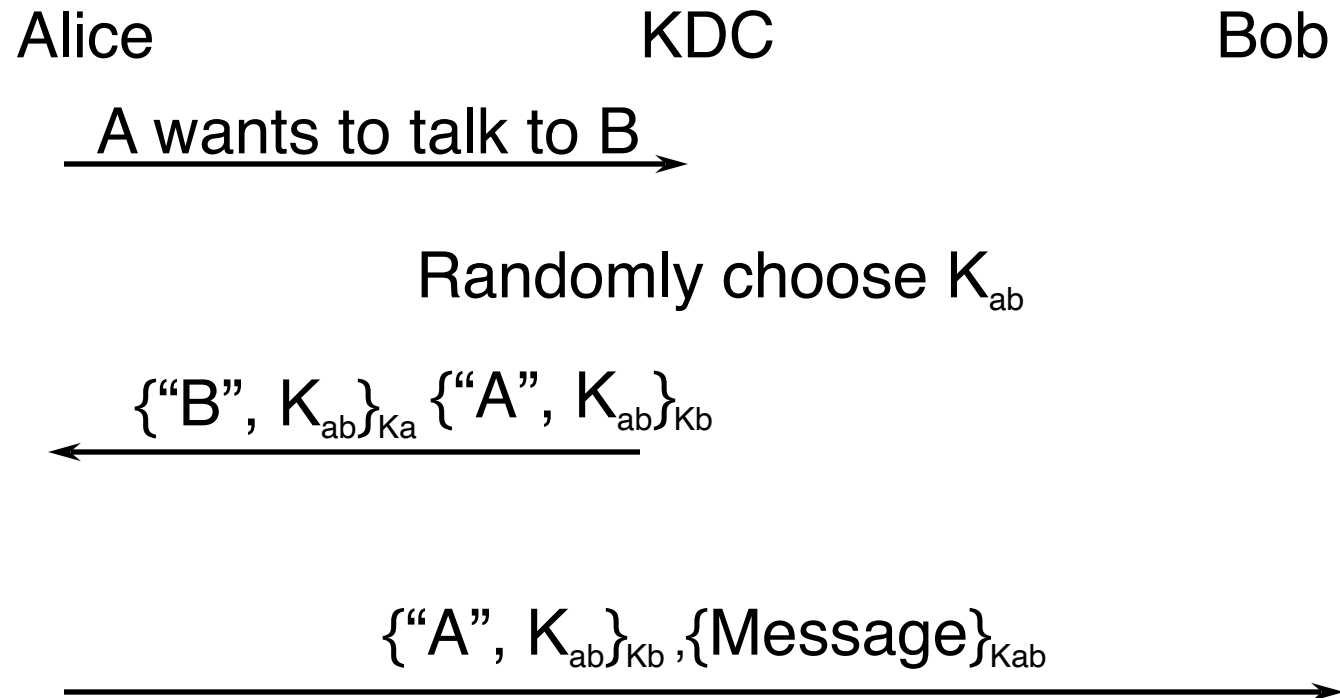
Carol
 K_C

How does Alice talk to Bob?

- She asks the KDC to introduce them



How it's actually done



Why this is better

- The KDC doesn't have to bother talking to Bob
- $\{\text{"A"}, K_{ab}\}_{K_b}$ is known as Alice's "ticket to Bob"
- Kerberos is still widely in use, because it is embedded in Microsoft's "Active Directory" service

Revocation

What if Alice discovers her private key is stolen

- Certificates have expiration dates
- But that would be months...the attacker can do a lot of damage in a few hours
- So, there should be a way for Alice to warn the CA that signed her certificate, and the CA should be able to invalidate her certificate
- This is known as a revocation mechanism
- There are many alternative designs for revocation mechanisms partially deployed

Very quick mention of
quantum

What is a quantum computer?

- Common misconception: Any program that runs on a classical computer would run a zillion time faster on a quantum computer
- Absolutely false!
- A quantum computer is not faster, it's just very different
- There are a very narrow set of algorithms that would run faster on a quantum computer
- The most important (to cryptography) is known as Shor's algorithm
- If there were a big enough quantum computer, Shor's algorithm could break all of our currently deployed public key algorithms (RSA, ECC, Diffie-Hellman)
- That would be very bad

Why we need to convert soon

- Note, quantum only threatens public key operations, not hashes and secret key (other than, for subtle reasons, the industry has moved to 256-bit secret keys and 512-bit hashes)
- It takes a long time to convert, and we can't predict when (or even if) a quantum computer big enough to threaten the key sizes in our currently deployed public key algorithms might exist
- There will be compliance rules that will require no use of current public key algorithms, and the perception by customers that those algorithms are not secure
- The “record now, decrypt later” issue: If someone records an entire https session (including the use of public keys to establish a secret key), then the conversation could be broken later

“Post-quantum” algorithms

- A very misleading name; should have been called quantum-safe algorithms
- These new algorithms are normal classical algorithms that work on normal classical computers
- They are public key algorithms that can replace RSA, ECC, and Diffie-Hellman, and are (hopefully) secure against both quantum and classical computers
- Why doesn't the world just pick one “best” one?
- Because they are all so different, with different tradeoffs of key size, signature size, computation required, etc.
- There are several that have been standardized, and the industry is busily migrating to these new algorithms

Thank You!