Security and Privacy in IoT
CSE 708 Fall 2021

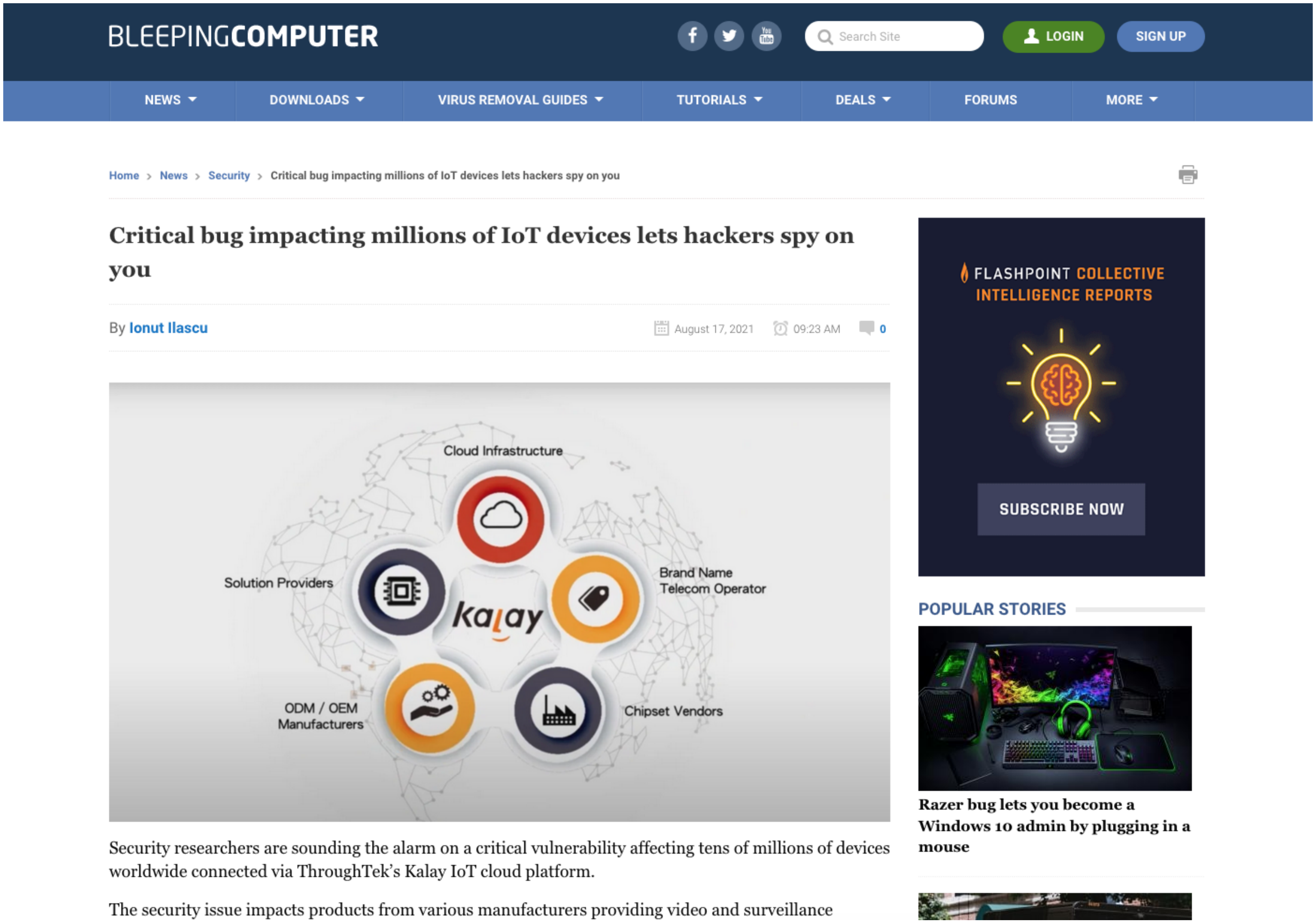
Fundamental Security and Cryptography Concepts

Department of Computer Science and Engineering
University at Buffalo

IoT and Security

- Why do we talk about Internet of Things and security?

IoT Security Breaches



BLEEPINGCOMPUTER

LOGIN SIGN UP

NEWS DOWNLOADS VIRUS REMOVAL GUIDES TUTORIALS DEALS FORUMS MORE

Home > News > Security > Critical bug impacting millions of IoT devices lets hackers spy on you

Critical bug impacting millions of IoT devices lets hackers spy on you

By Ionut Ilascu August 17, 2021 09:23 AM 0

Security researchers are sounding the alarm on a critical vulnerability affecting tens of millions of devices worldwide connected via ThroughTek's Kalay IoT cloud platform.

The security issue impacts products from various manufacturers providing video and surveillance

POPULAR STORIES

Razer bug lets you become a Windows 10 admin by plugging in a mouse

IoT Security Breaches

WIRED BACKCHANNEL BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY SIGN IN SUBSCRIBE

LILY HAY NEWMAN SECURITY NEWS 08.17.2021 08:00 AM

Millions of Web Camera and Baby Monitor Feeds Are Exposed

A vulnerability in the Kalay platform leaves countless IoT devices susceptible to hackers.

IoT Security Breaches

threatpost

Cloud Security / Malware / Vulnerabilities / InfoSec Insiders / Podcasts

Search

TeaBot Trojan Targets Banks via Hijacked Android Handsets

Gig Workers Being Paid $500 for Payroll Passwords

'FragAttacks': Wi-Fi Bugs Affect Millions of Devices

Wi-Fi devices going back to 1997 are vulnerable to attackers who can

INFOSEC INSIDER

Effective Threat-Hunting Queries in a Redacted World
August 24, 2021

Managing Privileged Access to Secure the Post-COVID Perimeter
August 23, 2021

How Ready Are You for a Ransomware Attack?
August 19, 2021

Kerberos Authentication Spoofing: Don't Bypass the Spec
August 18, 2021

The Overlooked Security Risks of The Cloud
August 17, 2021

Newsletter

Subscribe to *Threatpost Today*

Join thousands of people who receive the latest breaking cybersecurity news every day.

Subscribe now

Twitter

#Triada malware, pernicious and persistent, has resurfaced. Its most

IoT Security Breaches

KIMKOMANDO® 400+ radio stations in the USA and on demand
Tech advice you can trust™

WATCH NOW | LISTEN NOW | GET NEWSLETTERS | EMAIL KIM | LOGIN/JOIN THE COMMUNITY

NEWS | HOW-TOS | VIDEOS | SHOPPING | REVIEWS | KIM'S SHOW | FIND A STATION | PODCASTS

Search ...

SECURITY & PRIVACY

Warning: Bug allows complete takeover of dozens of popular routers

BY JAMES GELINAS, KOMANDO.COM • JUNE 20, 2020 SHARE:

Your router plays multiple roles in keeping your home network together. Not only does it connect all of your

Security Objectives

- Fundamental security objectives
  - Confidentiality (C): confidential or private information is not disclosed or made available to unauthorized parties
  - Integrity (I) : unauthorized modification of data is not permitted
  - Availability (A): resources are promptly available to authorized parties
- Confidentiality covers data confidentiality and privacy
- Integrity covers data integrity and system integrity

More on Security Objectives

- Other security concepts
  - Authenticity: the property of being genuine and being able to be verified and trusted
    - entity authentication: the entity is who it claims it is
    - data authentication: the data is coming from a trusted source
  - Access control: only authorized parties can use specific resources in compliance with their privileges
  - Non-repudiability (repudiability): inability (ability) to deny communication or actions
  - Accountability: the requirement that all actions of an entity are traced uniquely to that entity
    - covers non-repudiability, intrusion detection, fault isolation, etc.

Symmetric Encryption

- A computationally secure symmetric key encryption scheme is defined as:
  - a private-key encryption scheme consists of polynomial-time algorithms (Gen, Enc, Dec) such that
    1. Gen: on input the security parameter 1^n, outputs key k
    2. Enc: on input a key k and a message $m \in \{0,1\}^*$, outputs ciphertext c
    3. Dec: on input a key k and ciphertext c, outputs plaintext m (or fails)
  - we write $k \leftarrow \mathsf{Gen}(1^n)$, $c \leftarrow \mathsf{Enc}_k(m)$, and $m := \mathsf{Dec}_k(c)$
    - this notation means that Gen and Enc are probabilistic and Dec is deterministic

Symmetric Encryption

- Types of attacks
  - ciphertext only attack: adversary knows a number of ciphertexts
  - known plaintext attack: adversary knows some pairs of ciphertexts and corresponding plaintexts
  - chosen plaintext attack: adversary knows ciphertexts for messages of its choice
  - chosen ciphertext attack: adversary knows plaintexts for ciphertexts of its choice
- A standard minimum expected security is indistinguishable encryption under a chosen plaintext attack

Symmetric Encryption

- Symmetric encryption today would be instantiated with AES (Advanced Encryption Standard)
  - must use one of the secure encryption modes
  - a secure authenticated encryption mode can be used if confidentiality and integrity are simultaneously desired

Message Authentication Codes

- A MAC scheme is defined by three algorithms:
  - key generation: a randomized algorithm, which on input a security parameter 1^n, produces key a k
  - MAC generation: a possibly randomized algorithm, which on input a message m and key k, produces a tag t
  - MAC verification: a deterministic algorithm, which on input a message m, tag t, and key k, outputs a bit b

Message Authentication Codes

- We desire for a MAC to be existentially unforgeable under an adaptive chosen-message attack
  - an adversary is allowed to query tags on messages of its choice
  - at some point it outputs a pair (m, t)
  - the forgery is considered successful if m hasn't been queried before and t is a valid tag for it
  - as with encryption, security guarantees depend on the security parameter
- The most popular MAC instantiation is HMAC

Hash Functions

- A hash function h is an efficiently-computable function that maps an input x of an arbitrary length to a (short) fixed-length output $h(x)$

- h must satisfy the following security properties:

  – Preimage resistance (one-way): given $h(x)$, it is difficult to find x

  – Second preimage resistance (weak collision resistance): given x, it is difficult to find x' such that $x' \neq x$ and $h(x') = h(x)$

  – Collision resistance (strong collision resistance): it is difficult to find any x, x' such that $x' \neq x$ and $h(x') = h(x)$

Hash Functions

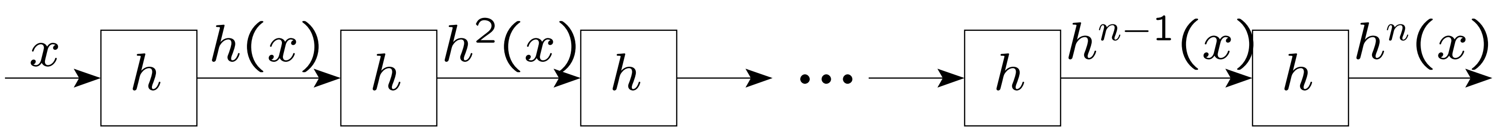
- Generic brute force attacks on hash functions with n-bit output have the following complexity
  - difficulty of finding a preimage is 2^n
  - difficulty of finding a second preimage is 2^n
  - difficulty of finding a collision with at least 50% probability is about $2^{n/2}$
  - all properties are desired for a general-use hash function
- Today a hash function is instantiated with SHA-2 (SHA-256 or higher) or SHA-3

Other Uses of Hash Functions

- Hash Chains
  - a method for authenticating multiple user logins or packet streams
  - consists of successive application of a hash function to a string
  - n applications of the hash function on x is denoted by $h^n(x)$
  - this produces a hash chain of length n
- Example:
  - $h^4(x) = h(h(h(h(x))))$ produces a hash chain of length 4

Hash Chains

- Authentication using hash chains
  - user generates a hash chain of length n
  - at time 1, the user sends $auth_1 = h^n(x)$ (and possibly authenticates it through other means)
  - the recipient stores $auth = auth_1$
  - at time 2, the user sends $auth_2 = h^{n-1}(x)$
  - the recipient checks whether $h(auth_2) = auth_1$ and, if so, accepts
  - the recipient updated $auth = auth_2$
  - etc.

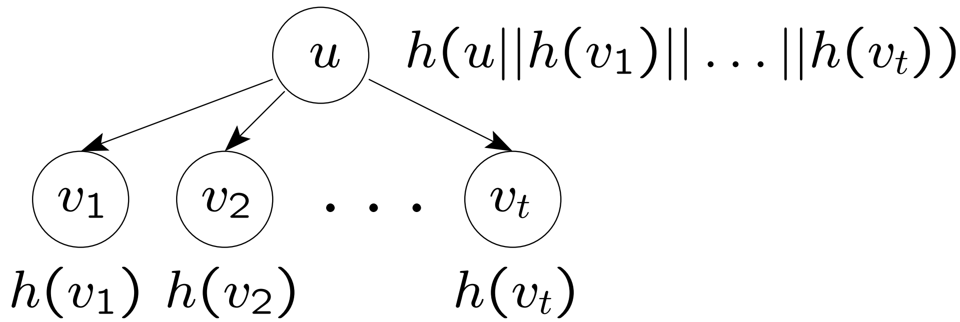


Merkle Hash Trees

- Merkle Hash Tree
  - integrity verification mechanism for hierarchically structured documents or databases
  - the technique works on trees only
  - the hash of the tree is computed in the bottom-up fashion
- Generation of a Merkle hash tree
  - for a leaf node v, simply compute its hash $h(v)$
  - for a non-leaf node u with children $v_1, \ldots, v_t$, compute its hash as $h(u||h(v_1)||\ldots||h(v_t))$

Merkle Hash Trees

- Merkle Hash Tree



  – this computation continues until the hash of the root is computed

  – the hash of the root corresponds to the hash of the entire tree

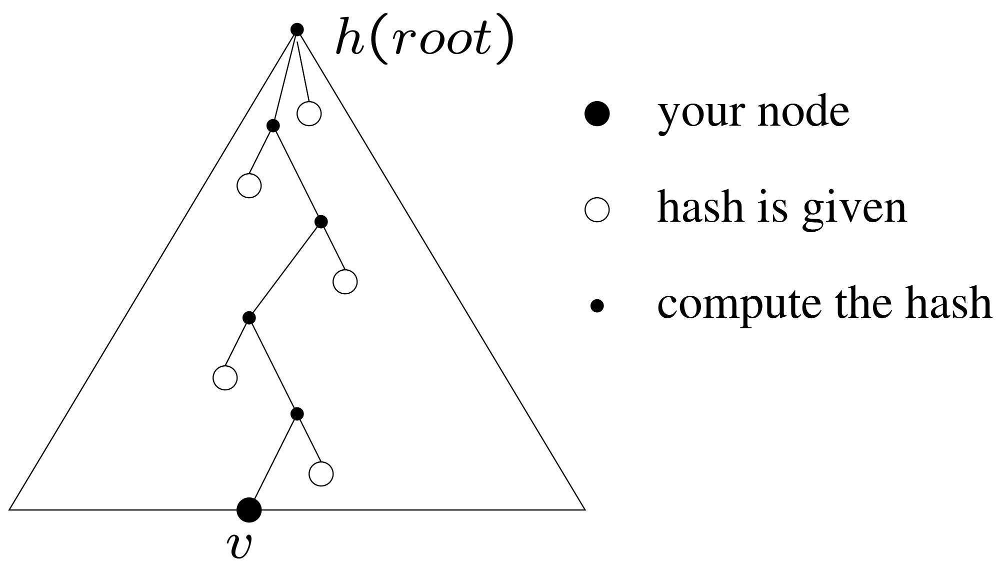
- Integrity verification

  – node integrity verification is much faster than hashing the entire tree

  – to check node v, obtain hashes of the nodes on the path from v to the root

Merkle Hash Trees

- Integrity verification in Merkle Hash Tree



  - compute the hash of v and combine it with other hashes on the path to the root
  - compare your hash of the root with what you are given
  - the node you are authenticating doesn't have to be a leaf

Pseudorandom Generator

- Let G be a (deterministic) algorithm that on input n-bit string s outputs a string of length $\ell(n)$
- G is a pseudorandom generator if the following is true:
  1. (expansion) for any n, output is longer than input: $\ell(n) > n$
  2. (pseudorandomness) any PPT distinguisher D can't tell the difference with non-negligible probability:

     $$|\mathsf{Pr}[D(r) = 1] - \mathsf{Pr}[D(G(s)) = 1]| \leq \mathsf{negl}(n)$$

     where r and s are random strings of size $\ell(n)$ and n
- The seed s must be treated similar to a key

Pseudorandom Function

- An efficient function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a pseudorandom function if any PPT distinguisher D cannot tell apart outputs of F_k and f, i.e.,

$$|\Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1]| \leq \mathsf{negl}(n)$$

  for a uniformly chosen function $f : \{0,1\}^n \to \{0,1\}^n$ and uniformly chosen key $k \leftarrow \{0,1\}^n$

- A typical instantiation of a PRF is AES

- A PRF can also be used to build a PRG
  - define $\mathsf{PRG}(k) := \mathsf{PRF}_k(0)||\mathsf{PRF}_k(1)||\ldots$

Public Key Encryption

- A public-key encryption scheme consists of three algorithms (Gen, Enc, Dec) such that:

  1. key generation Gen, on input security parameter 1^n, outputs a public-private key pair (pk, sk)
  2. encryption Enc, on input public key pk and messages m from the message space, outputs ciphertext $c \leftarrow \mathsf{Enc}_{pk}(m)$
     - message space often depends on pk
  3. decryption Dec, on input private key sk and ciphertext c, outputs a message $m := \mathsf{Dec}_{sk}(c)$ or a special failure symbol $\perp$.

- As before, the minimum security expectation is indistinguishability under a chosen-plaintext attack

Digital Signatures

- A signature scheme is defined by three algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ such that:

  1. key generation algorithm Gen, on input a security parameter 1^n, outputs a key pair (pk, sk), where pk is the public key and sk is the private key.

  2. signing algorithm Sign, on input a private key sk and message $m \in \{0, 1\}^*$, outputs a signature σ, i.e., $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$

  3. verification algorithm Vrfy, on input a public key pk, a message m, and a signature σ, outputs a bit b, where $b = 1$ means the signature is valid and $b = 0$ means it is invalid, i.e., $b := \mathsf{Vrfy}_{pk}(m, \sigma)$

- We'll want to achieve the same level of security as for MACs: existential unforgeability under an adaptive chosen-message attack

Groups

- A group G is a set of elements together with a binary operation $\circ$ such that
  - the set is closed under the operation $\circ$, i.e., for every $a, b \in G$, $a \circ b$ is a unique element of G
  - the associative law holds, i.e., for all $a, b, c \in G$, $a \circ (b \circ c) = (a \circ b) \circ c$
  - the set has a unique identity element e such that $a \circ e = e \circ a = a$ for every $a \in G$
  - every element has a unique inverse a^{-1} in G such that $a \circ a^{-1} = a^{-1} \circ a = e$

Groups

- Size of a group
  - a group is finite if it has only a finite number of elements
  - the number of elements of a finite group is called the order of the group
- The multiplicative group modulo m is denoted by $\mathbb{Z}_m^*$
- A cyclic group is one that contains an element a whose powers (using multiplicative notation of group operation) a^i and a^{-i} make up the entire group
- An element a with such property is called a generator of the group

Discrete Logarithm Problem

- Discrete logarithms
  - we are given a cyclic group G of order q
  - then there exists an element $g \in G$ such that $G = \langle g \rangle = \{g^i : 0 \leq i \leq q - 1\}$
  - for each $h \in G$ there is a unique x such that $g^x = h$
  - such x is called the discrete logarithm of h with respect to g and we use $x = \log_g h$
- The discrete logarithm problem
  - in a cyclic group G with given generator g, compute unique $\log_g h$ for a random element $h \in G$

Discrete Logarithm Problem

- Groups in which the discrete logarithm problem is hard
  - multiplicative group over $\mathbb{Z}_p^*$ with prime p with certain constraints on the order of the group
  - a subgroup of the above
    - this will allow us to produce a group of prime order q
  - an elliptic curve group modulo a prime p

Diffie-Hellman Key Exchange

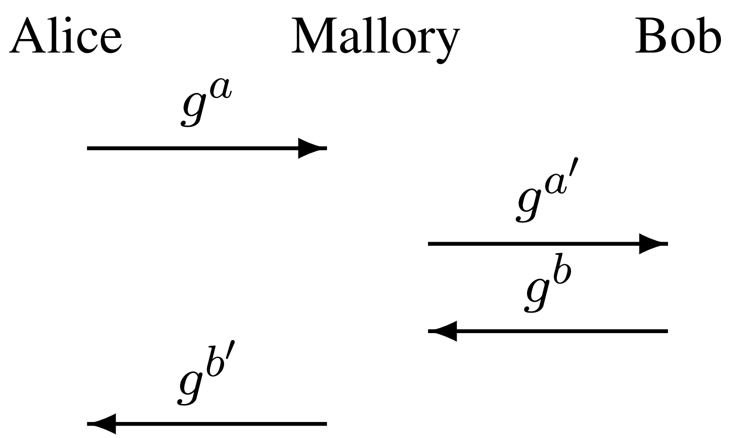
- Diffie-Hellman key exchange protocol
  - Alice and Bob want to compute a shared key unknown to eavesdroppers
  - Alice and Bob share public parameters: a group G of order q and a generator g
  - Alice randomly chooses $x \in \mathbb{Z}_q$ and sends g^x to Bob: $A \xrightarrow{g^x} B$
  - Bob randomly chooses $y \in \mathbb{Z}_q$ and sends g^y to Alice: $A \xleftarrow{g^y} B$
  - the shared secret is set to g^{xy}
    - Alice computes it as $(g^y)^x = g^{xy}$
    - Bob computes it as $(g^x)^y = g^{xy}$
  - it is believed to be infeasible for an eavesdropper to compute g^{xy} given g^x and g^y

Diffie-Hellman Key Exchange

- Diffie-Hellman key exchange protocol
  - Alice and Bob are able to establish a shared secret with no prior relationship
  - it is believed to be infeasible for an eavesdropper to compute g^{xy} given g^x and g^y
- Diffie-Hellman problem
  - Computational Diffie-Hellman (CDH) problem
    - given g, g^x and g^y, compute g^{xy}
  - Decision Diffie-Hellman (DDH) problem
    - given g, g^x, g^y, and g^z, determine whether $xy = z$ (modulo q)

Diffie-Hellman Key Exchange

- Man-in-the-middle attack on Diffie-Hellman key exchange



  - Alice shares the key $g^{ab'}$ with Mallory
  - Bob shares the key $g^{a'b}$ with Mallory
  - Alice and Bob do not share any key

- A solution is to build an authenticated Diffie-Hellman key exchange

Diffie-Hellman Key Exchange

- Certificates can be used to aid authentication
  - each user U has a private signing key sk_U and the corresponding public verification key pk_U
  - there is a trusted authority TA that signs keys
  - user U holds a certificate $\mathsf{cert}(U)$ issued by the TA

$$\mathsf{cert}(U) = (U, pk_U, \sigma_{TA}(U, pk_U))$$

- Signatures and certificates can be used to strengthen Diffie-Hellman key exchange
  - different versions of authenticated Diffie-Hellman key exchange are used including in TLS

Bilinear Maps

- A one-way function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map if the following conditions hold:
  - (Efficient) $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are groups of the same prime order p, and there exists an efficient algorithm for computing e.
  - (Bilinear) For all $g \in \mathbb{G}_1$, $\tilde{g} \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$, $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{ab}$.
  - (Non-degenerate) If g generates $\mathbb{G}_1$ and $\tilde{g}$ generates $\mathbb{G}_2$, then $e(g, \tilde{g})$ generates $\mathbb{G}_T$.
- Bilinear maps are also called groups with pairings

Commitments

- Commitment schemes
  - a commitment scheme allows one to "commit" to a message m by computing a committed value com
  - it can later be opened to reveal m
  - the following properties are required to hold:
    - hiding property: commitment com reveals nothing about message m
    - binding property: it is infeasible to find another message $m' \neq m$ such that com can be opened to m'

Commitments

- A commitment scheme is defined by three algorithms
  - Gen: randomized algorithm that takes a security parameter 1^n and outputs public parameters params
  - Com: randomized algorithm that takes params and a message $m \in \{0,1\}^n$ and outputs commitment com
    - we make the randomness that Com uses explicit, denote it by r, and use $\mathsf{com} = \mathsf{Com}(\mathsf{param}, m, r)$
  - Open: a deterministic algorithm that decommits to m by typically disclosing m and r
    - the verifier that check whether com is in fact equal to $\mathsf{Com}(\mathsf{params}, m, r)$

Commitments

- We can use hash functions to create a commitment scheme (in the random oracle model):
  - Gen takes a security prameter 1^n and chooses an appropriate hash funcion h
  - to commit to m, choose uniform $r \in \{0,1\}^n$ and output $\mathsf{Com}(m,r) := h(m||r)$
  - hiding follows because adversary can query $h(*||r)$ with only a negligible probability
  - binding follows from the collision resistance property of h
- A popular number-theoretic commitment is Pedersen commitment of the form $\mathsf{Com}(m,r) = g^m h^r$ in a DDH group

Homomorphic Encryption

- Homomorphic encryption allows for computing on encrypted data without access to the underlying plaintexts
  - it is a special type of encryption that, given ciphertexts, permits computation on the underlying plaintexts

$$\mathsf{Enc}_k(m_1) \otimes \mathsf{Enc}_k(m_2) = \mathsf{Enc}_k(m_1 \oplus m_2)$$

  - homomorphic encryption enables computation on encrypted data and results in efficient protocols for certain problems
  - besides Gen, Enc, and Dec, additional algorithm(s) specify how to use homomorphic properties

Homomorphic Encryption

- We'll look at two types of public-key homomorphic encryption
- The first type is called partially homomorphic encryption (or just HE for short) and comes with one homomorphic operation
  - of most significant importance to us is the ability to add (integer) values inside ciphertexts
  - we have $\mathsf{Enc}_{pk}(m_1) \cdot \mathsf{Enc}_{pk}(m_2) = \mathsf{Enc}_{pk}(m_1 + m_2)$
  - which in turn implies $\mathsf{Enc}_{pk}(m)^c = \mathsf{Enc}_{pk}(m \cdot c)$
  - Paillier encryption scheme (1999) is a popular cryptosystem of this type

Homomorphic Encryption

- The second type is called fully homomorphic encryption (FHE)
  - it supports two types of operations on ciphertexts: addition and multiplication
  - this type enables any function to be evaluated on encrypted data
  - this is suitable for secure computation outsourcing to a single server
- The drawback of FHE is its speed
  - it is not always suitable for moderate or large functions or amounts of data

Summary

- There are many different types of tools which can be used to build secure solutions
- We'll explore them as part of this course