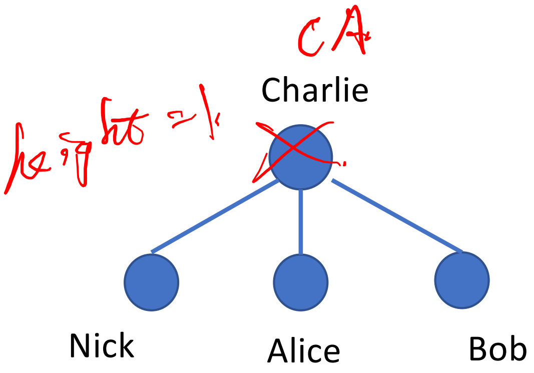
Obtaining a User Certificate

The Trusted Directory

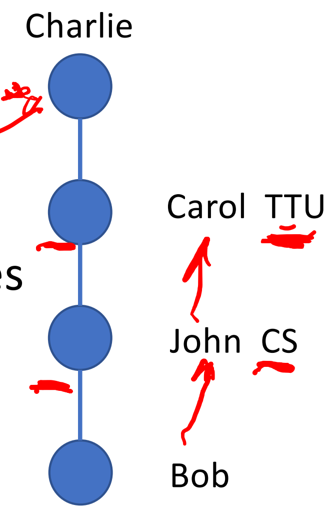
- Idea: Make a central, trusted directory (TD) from where you can fetch anybody's public key
  - The TD has a public/private keypair PK_{TD}, SK_{TD}
  - The directory publishes PK_{TD} so that everyone knows it (baked into computers, phones, OS, etc.)
  - When you request Bob's public key, the directory sends a certificate for Bob's public key
    - {"Bob's public key is PK_B"}$_{SK_{TD}^{-1}}$
  - If you trust the directory, then now you trust every public key from the directory
  - The ~~directory itself~~ is not responsible for the ~~creation of public~~ keys. It merely provides an easily accessible location for users to obtain certificate. CA
- What do we have to trust? (assumption) Bob
  - TD won't sign a key without verifying the identity of the owner
  - We have received TD's key correctly

The Trusted Directory



- Let’s say that Charlie (an authority) runs the TD
  - We want Nick’s public key: Ask Charlie
  - We want Alice’s public key: Ask Charlie
  - We want Bob’s public key: Ask Charlie
  - Charlie has better things to do (like making sure his private key isn’t stolen)!
- Problems?
  - Scalability: One directory won’t have enough compute power to serve the entire world
  - Single point of failure:
    - If the directory fails, your application might become unavailable
    - If the directory is compromised, you can’t trust anyone
    - If the directory is compromised, it is difficult to recover

Certificate Authorities



- Addressing scalability: Hierarchical trust
  - The roots of trust may **delegate** trust and signing power to other authorities
    - {"Carol Christ's public key is PK_{CC}, and I trust her to sign for TTU"}$_{SK_{C}^{-1}}$
    - {"John Canny's public key is PK_{JC}, and I trust him to sign for the CS department"}$_{SK_{CC}^{-1}}$
    - {"Bob's public key is PK_{Bob} (but I don't trust him to sign for anyone else)"}$_{SK_{JC}^{-1}}$
  - Charlie is still the root of trust (**root certificate authority**, or **root CA**)
  - CC and JC receive delegated trust (**intermediate CAs**)
  - Bob's identity can be trusted
- Addressing scalability: Multiple trust anchors
  - There are ~150 root CAs who are implicitly trusted by most devices
  - Public keys are hard-coded into operating systems and devices

Hierarchical CAs

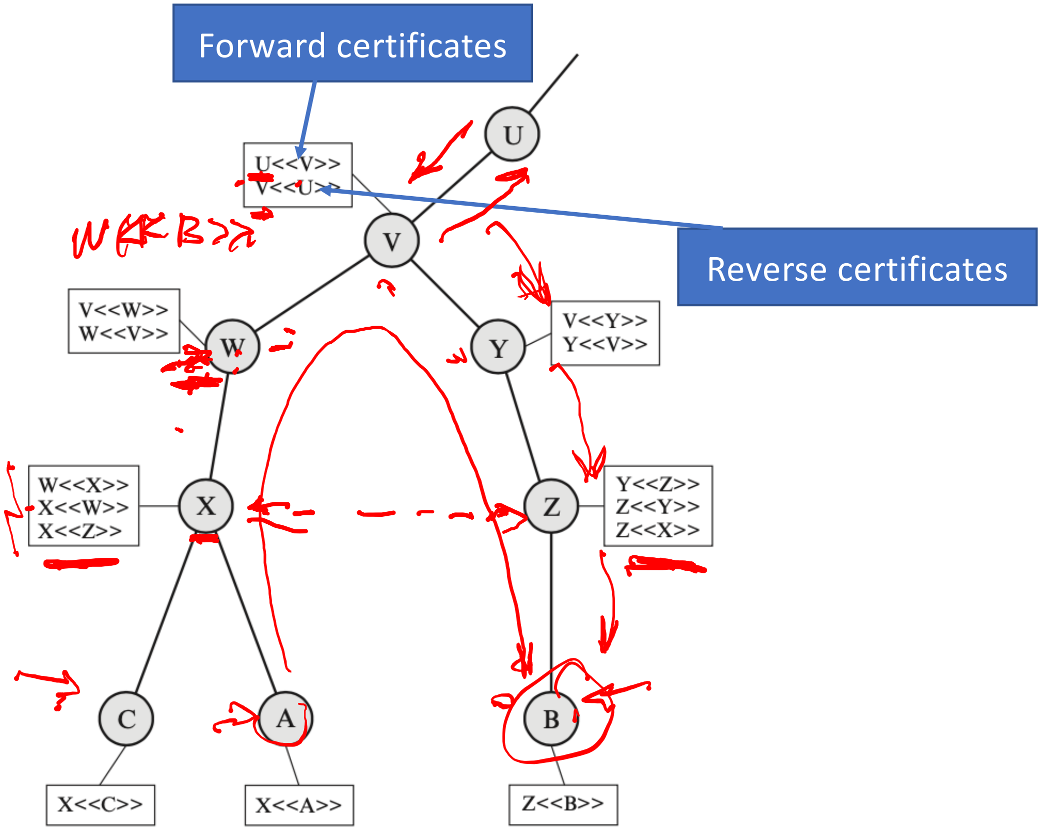
if W cached ~~cert~~ PK of B then 2,

- A acquires B’s public key

$X \ll W \gg W \ll V \gg V \ll Y \gg Y \ll Z \gg Z \ll B \gg$

1 2 3 4 5

When Alice wants to request Bob’s PK, she can obtain this list of certificates (certificate chain) from any untrusted service, and does not need to contact the respective parties because the signatures are unforgeable



Certificate Revocation

Revocation

- What happens if a certificate expires, or authority issues a new certificate?
- 🔍 **Expiration ≠ Revocation**
  - **Expiration** means the certificate has *naturally reached the end of its validity period.*
    → It's no longer trusted **after** that date.
  - **Revocation** means the certificate was **explicitly invalidated early** — before its expiration — because something went wrong.
- So, a certificate can still look *valid by date*, but actually be **dangerous** to trust.

Why a certificate might be revoked?

- There are several reasons a CA might revoke a certificate before it expires:

Reason	Description
🔑 **Private key compromised**	The website’s secret key got leaked or stolen.
🧑‍💻 **Owner no longer controls the domain**	e.g., domain ownership changed.
🧾 **CA issued it by mistake**	Misconfiguration or policy violation.
🚫 **Malicious activity detected**	The certificate is being misused for phishing or MITM.

- In any of these cases, the certificate’s expiration date is irrelevant — it must be rejected **immediately**.

Revocation: Expiration Dates

- Approach I: Each certificate has an expiration date
  - As a certificate nears expiration or is compromised, the owner must request a new one from the CA
  - Once expired, a compromised or invalid certificate automatically becomes unusable.
- Benefits
  - Mitigates damage: Eventually, the bad certificate will become harmless
- Drawbacks
  - Adds management burden: Everybody has to renew their certificates frequently
  - If someone forgets to renew a certificate, their website might stop working
- Tradeoff: How often should certificates be renewed?
  - Frequent renewal: More secure, less usable – adds maintenance complexity
  - Infrequent renewal: Less secure, more usable – extends the risk window for compromised certificates
  - *Let*'*s Encrypt* (a certificate authority) chose very frequent renewal before 2022
    - Let's Encrypt is a non-profit certificate authority run by Internet Security Research Group that provides X.509 certificates for Transport Layer Security encryption at no charge.