Phishing Attacks

**Phishing attack:** User clicks link in email and visits attacker’s web site, believing it is a legitimate web site.

**Spear-phishing attack:** User is misled because email contains PII or other information only known to a trusted individual.

**Solution Idea:** User U’s authentication credentials include a secret known to U and the site name that is intended relying party.
MIM attack for Duo

A → Bad: login as A
Bad → Good: login as A
Good → A’s duo: OK?
A’s duo: Yes (thinking request is from Good)
... Good thinks Bad is user A.
... Bad can impersonate user A.

Notice: Login credentials do not include identity “Good”.
A User’s Authentication Token

Token:
- Can communicate with user’s nearby laptop/desktop
- Has a button so user can confirm intent
- Has a small memory
  - Stores a token-unique secret \( \text{sec} \)
- Has crypto hardware:
  - Can generate fresh private/public key pair on demand
  - Can generate AES key for source from \( f(\text{sec}, \text{source}) \)
  - Can encrypt/decrypt AES
  - Can digitally sign with private key
Overall Architecture

- Uses a different public/private key pair for each site and each user.
- Token uses different AES key for each site (key is based on secret sec and site name S).
  - A user U’s credentials for a site S is a pair:
    - $K(U,S)$-Enc( pub/priv key pair), pub key
- Relying party sends to token a challenge requesting signature to “prove” token knows user’s private key for site.
  - Site S sends challenge to U’s browser:
    - S includes with challenge U’s credentials for site S
  - U’s browser forwards: name S and U’s credentials for site S.
  - U’s token derives AES key to extract private key for site S
User Enrollment for Site S

Site S requests *authenticator* from browser.
U’s browser requests authenticator from token.

Token does:
- Generate fresh pub/priv key $PK_S / pk_S$ for site S
- Generate AES key $K_S$ from token’s secret and name $S$
- Generate authenticator for U at S. It includes:
  - $K_S$-$\text{Enc}(PK_S / pk_S)$
  - $PK_S$
- Send authenticator to browser and forget $K_S$, $PK_S$, $pk_S$

Browser sends authenticator to site S

Site S stores authenticator with info for user U

Note: Authenticator might be stored as a cookie at browser. Cookie would be encrypted using site S local key. Cookie would be sent to S whenever U visits S.
User Authentication at S

Site S sends to browser:
- authenticator for U
- fresh challenge r

Browser forwards to token
Token reconstructs AES key $K_S$ from token’s secret and name S
- Only name of actual site S will work --- phishing site will have a different name.

Token extracts private key $pk_S$ from authenticator
Token asks user’s consent to proceed
Token signs challenge r using $pk_S$
Token sends signed challenge to browser
Browser forwards signed challenge to site S
Site S checks signature using $PK_S$ from authenticator.