Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Enkratni digitalni podpisi in njihova uporaba

· Kriptografija in teorija kodiranja 2 ·

Marko Košmerl
Ljubljana, julij 2011

Kazalo

1 Uvod

Digitalni podpisi za enkratno uporabo so namenjeni podpisovanju kvečjemu enega elektronskega sporočila nato pa je za nov podpis potrebna menjava javnega in zasebnega ključa, medtem ko v nasprotnem primeru podpis ni več varen.
Sam koncept je znan že relativno dolgo pa vendarle je bil do nedavnega predmet študije zgolj zaradi teoretične vrednosti. Prvič sta ga leta 1979 neodivsno predstavila Leslie Lamport in Michael O. Rabin.

1.1 Motivacija

Nedavno so se na tem področju pojavile nove sheme do katerih je prišlo zaradi potreb po hitrih in računsko poceni načinov overjanja v IP multicast/broadcast in ad-hoc omrežjih. Pri problemu overjanja broadcast prometa je cilj odpraviti nevarnosti kot so zlonamerno spreminjanje ali brisanje toka podatkov medtem, ko je v ad-hoc omrežjih cilj overiti vozlišča in s tem zavarovati omrežje pred zlonamernimi vozlišči.

1.2 Cilj in zgradba projektne naloge

Cilj te naloge je predstaviti sam koncept enkratnih digitalnih podpisov ter opisati in primerjati nekaj nedavno objavljenih shem, ki so bile predlagane kot rešitev na področjih, kjer je hitro overjanje in preverjanje integritete ključnega pomena.

V naslednjem poglavju bomo najprej pogledali definicijo digitalnega podpisa nato pa spoznali shemi BiBa, HORS ter shemi Shema 1 in Shema 2, ki sta izboljšana verzija sheme HORS. V poglavju 3 bomo sheme temeljito analizirali in jih primerjali za konec pa v poglavju 4 predstavili kako s pomočjo predstavljenih shem rešimo problem overjanja podatkov v broadcast omrežjih ter problem varnosti usmerjanja v ad-hoc omrežjih.

2 Sheme za enkratno podpisovanje

Poglejmo si sedaj definicijo nato pa način generiranja ključev in samo metodo podpisovanja in preverjanja pristnosti sporočila izbranih shem.

2.1 Definicija digitalnega podpisa

Samo klasifikacijo digitalnih podpisov najde bralec npr. na prosojnicah [2], za boljši vpogled v digitalne podpise prav tako priporočam tudi članek Zaupati ali ne zaupati [3].

Naj za uvod navedemo definicijo in še enkrat omenimo, da se mora pri novem podpisu sporočila pri shemah za enkratno uporabo, zasebni in pripadajoči javni ključ znova generirati.

Shema za digitalno podpisovanje je peterka (P, A, K, S, V), za katero velja, da je P je končna množica sporočil, A končna množica podpisov in K končna množica ključev. Za vsak ključ $k \in K$ obstaja algoritem za podpisovanje

$$\text{sig}_k \in S, \text{sig}_k : P \to A$$

in algoritem za preverjanje podpisa

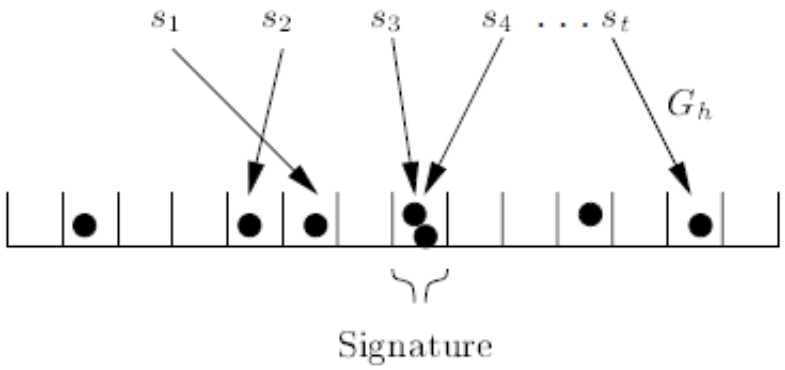
$$\text{ver}_k \in V, \text{ver}_k : P \times A \to \{\text{true}, \text{false}\}$$

Prav tako mora veljati $\text{ver}_k(x, y) = \begin{cases} \text{true}, & \text{če } y = \text{sig}_k(x); \\ \text{false}, & \text{če } y \neq \text{sig}_k(x); \end{cases}$

Uporabljali bomo naslednjo notacijo. Naj bo $f : \{0,1\}^\ell \to \{0,1\}^\ell$ enosmerna funkcija, $H : \{0,1\}^* \to \{0,1\}^{k \log_2 t}$ enosmerna zgoščevalna funkcija ter $G_h : \{0,1\}^\ell \to [0, n-1]$ družina enosmernih zgoščevalnih funkcij v naključnem oraklu. Naključen orakel (angl. random oracle) je matematična abstrakcija in se ga navadno uporablja za poenostavljeno analizo varnosti. Nanj gledamo kot na črno škatlo, ki nam vrne naključen odgovor z enakomerno porazdelitvijo, prav tako pa na isto vprašanje odgovori vedno isto.

2.2 BiBa (Bins and Balls)

Shema BiBa [4, 5] temelji na paradoksu rojstnih dnevov. Podpisovalec ima t naključno izbranih števil, ki jih s pomočjo enosmerne zgoščevalne funkcije preslika v neko število množice $\{0, 1, ..., n-1\}$. Podpis tvorijo naključna števila, ki imajo enako sliko. Preprosto povedano, podpisovalec naključno meče t žog v n košar in na koncu za podpis vzame k žog, ki zaključijo v isti košari. Na sliki 1 vidimo, da je podpis par (s_3, s_4).



Slika 1: Koncept sheme BiBa

Algoritem 2.2.1 BiBa - Generiraj javni in zasebni ključ

Vhod: ℓ, k, t

Generiraj t naključnih besed dolžine ℓ bitov: $s_1, s_2, ..., s_t$

Naj bo $v_i = f(s_i)$ for $1 \leq i \leq t$

Izhod: JK $= (k, v_1, v_2, ..., v_t)$ in ZK $= (k, s_1, s_2, ..., s_t)$

Algoritem 2.2.2 BiBa - Podpisovanje

Vhod: Sporočilo m in zasebni ključ ZK $= (k, s_1, s_2, ..., s_t)$

$c = 0$

loop

$c = c + 1$

$h = H(m||c)$

$b_i = G_h(s_i)$ za vsak i, $1 \leq i \leq t$

if $\exists (i_1, i_2, ...i_k) : b_{i_1} = b_{i_2} = ... = b_{i_k}$ **then**

Končaj

end if

end loop

Izhod: $\sigma = (c, s_{i_1}, s_{i_2}, ..., s_{i_k})$

Algoritem 2.2.3 BiBa - Verifikacija

Vhod: Sporočilo m in podpis $\sigma = (c, s_{i_1}, s_{i_2}, ..., s_{i_k})$
  Odgovor = `Zavrni`
  if $s_{i_1} \neq s_{i_2} \neq ... \neq s_{i_k}$ **then**
    $h = H(m||c)$
    if $G_h(s_{i_1}) = ... = G_h(s_{i_k})$ **then**
      Odgovor = `Sprejmi`
    end if
  end if
Izhod: Odgovor.

Sama varnost sheme je lahko določena na več načinov, k-kratno trčenje žog je le ena izmed izbir. Lahko bi povečali število košev, vendar bi s tem povečali velikost javnega ključa. Prav tako bi lahko kot ključ uporabili k dvojnih trčenj. Zadnji način, ki ga omenimo, je uporaba večkrožne sheme pri kateri koši v katerih je prišlo do k_1-kratnih trčenj sodelujejo v drugem krogu kot žoge. Izkaže se, da so enokrožne sheme prav tako varne kot večkrožne.

2.3 HORS (Hash to obtain random subset)

Ideja sheme HORS [6] je, da preslikavo $H(m)$ razdelimo v k nizov dolžine $log_2 t$ bitov in vsak niz interpretiramo kot celo število. Dobimo torej naključno množico velikosti kvečjemu k.

Javni in zasebni ključ sheme HORS [6] se generirata po algoritmu 2.2.1.

Algoritem 2.3.1 HORS - Podpisovanje

Vhod: Sporočilo m in zasebni ključ ZK $= (k, s_1, s_2, ..., s_t)$
  $h = H(m)$
  Razdeli h v k podnizov $h_1, h_2, ..., h_k$, $|h_i| = \log_2 t$
  Interpretiraj h_j kot celo število i_j za $1 \leq j \leq k$
Izhod: $\sigma = (s_{i_1}, s_{i_2}, ..., s_{i_k})$

Algoritem 2.3.2 HORS - Verifikacija

Vhod: Sporočilo m, $\sigma = (s'_{i_1}, s'_{i_2}, ..., s'_{i_k})$ in JK $= (k, v_1, v_2, ..., v_t)$
$h = H(m)$
Razdeli h v k podnizov $h_1, h_2, ..., h_k$, $|h_i| = \log_2 t$
Interpretiraj h_j kot celo število i_j za $1 \leq j \leq k$
Izhod: `Sprejmi`, če za j, $1 \leq j \leq k$, $f(s'_j) = v_{i_j}$, drugače `zavrni`.

Da zadostimo potrebam varnosti mora veljati, da je neizvedljivo najti dva poljubna sporočila m_1 in m_2, da velja $H(m_1) \subseteq H(m_2)$. Ker smo H modelirali v naključnem oraklu, je zahteva iz same definicije modela izpolnjena.

2.4 Shema 1

Če se ozremo na zadnjo shemo - HORS, vidimo, da v primeru, če ima napadalec A za sporočilo m podpis SIG $= (s_{i_1}, ..., s_{i_k})$, lahko A ponaredi podpis za neko sporočilo s samim spreminanjem mest elementov v podpisu. Če lahko A najde m' ali m'' tako, da velja $H(m') = h_2||h_1||h_3||...||h_k$ ali $H(m'') = h_3||h_2||h_1||...||h_k$, lahko ustvari podpis $(s_{i_2}, s_{i_1}, s_{i_3}, ..., s_{i_k})$ za m' ali $(s_{i_3}, s_{i_2}, s_{i_1}, ..., s_{i_k})$ za m''. Za izboljšanje varnosti z namenom zmanjšati verjetnost uspeha omenjenega napada, se v shemi, ki jo poimenjujemo kar *Shema 1* [1], spremeni obliko javnega ključa in vstavi dodaten pogoj pri generiranju ter verifikaciji podpisa.

Algoritem 2.4.1 Shema 1 - Generiraj javni in zasebni ključ

Vhod: l, k, t
Generiraj t naključnih besed dolžine l bitov: $s_1, s_2, ..., s_t$
Naj bo $p_i = f(s_i)$, $v_i = f(p_i)$ for $1 \leq i \leq t$
Izhod: JK $= (v_1, v_2, ..., v_t)$ in ZK $= ((s_1, s_2, ..., s_t), (p_1, p_2, ..., p_t))$

Algoritem 2.4.2 Shema 1 - Podpisovanje

Vhod: Sporočilo m in zasebni ključ ZK $= (k, s_1, s_2, ..., s_t)$

loop

Izberi naključno vrednost c

$h = H(m||c)$ Razdeli h v k podnizov $h_1, h_2, ..., h_k$, $|h_i| = \log_2 t$

Interpretiraj h_j kot celo število i_j za $1 \leq j \leq k$

if $i_1 < ... < i_{\frac{k}{2}}$, $i_{\frac{k}{2}+1} < ... < i_k$, $\{i_1, ..., i_{\frac{k}{2}}\} \bigcap \{i_{\frac{k}{2}+1}, ..., i_k\} = \emptyset$ **then**

Končaj

end if

end loop

Izhod: $\sigma = (c, (s_{i_1}, ..., s_{i_{\frac{k}{2}}}), (p_{i_{\frac{k}{2}+1}}, ..., p_{i_k}))$

Algoritem 2.4.3 Shema 1 - Verifikacija

Vhod: Sporočilo m, $\sigma = (c', (u_{i_1}, ..., u_{i_{\frac{k}{2}}}), (u_{i_{\frac{k}{2}+1}}, ..., u_{i_k}))$ in JK

Odgovor = `Zavrni`

$h = H(m)$

Razdeli h v k podnizov $h'_1, h'_2, ..., h'_k$, $|h'_i| = \log_2 t$

Interpretiraj h'_j kot celo število i'_j za $1 \leq j \leq k$

if $i'_1 < ... < i'_{\frac{k}{2}}$, $i'_{\frac{k}{2}+1} < ... < i'_k$, $\{i'_1, ..., i'_{\frac{k}{2}}\} \bigcap \{i'_{\frac{k}{2}+1}, ..., i'_k\} = \emptyset$ **then**

if $f(f(u_j)) = v_{i'_j}$ in $f(u_{j+\frac{k}{2}}) = v_{i'_{j+\frac{k}{2}}}$, $1 \leq j \leq \frac{k}{2}$ **then**

Odgovor = `Sprejmi`

end if

end if

Izhod: Odgovor.

2.5 Shema 2

Cena podpisa *Sheme 1* je kot bomo videli v razdelku 3.1.3, precej draga. Da jo zmanjšamo, spremenimo pogoj podpisovanja in dobimo spodaj predstavljeno shemo poimenovano *Shema 2* [1].

Javni in zasebni ključ se generirata po algoritmu 2.4.1.

Algoritem 2.5.1 Shema 2 - Podpisovanje

Vhod: Sporočilo m in zasebni ključ ZK $= (k, s_1, s_2, ..., s_t)$
  loop
    Izberi naključno vrednost c
    $h = H(m||c)$ Razdeli h v k podnizov $h_1, h_2, ..., h_k$, $|h_i| = \log_2 t$
    Interpretiraj h_j kot celo število i_j za $1 \leq j \leq k$
    if $i_1 \neq i_2 \neq ... \neq i_k$ **then**
      Končaj
    end if
  end loop
Izhod: $\sigma = (c, (s_{i_1}, ..., s_{i_{\frac{k}{2}}}), (p_{i_{\frac{k}{2}+1}}, ..., p_{i_k}))$

Algoritem 2.5.2 Shema 2 - Verifikacija

Vhod: Sporočilo m, $\sigma = (c', (u_{i_1}, ..., u_{i_{\frac{k}{2}}}), (u_{i_{\frac{k}{2}+1}}, ..., u_{i_k}))$ in JK
  Odgovor = `Zavrni`
  $h = H(m)$
  Razdeli h v k podnizov $h'_1, h'_2, ..., h'_k$, $|h'_i| = \log_2 t$
  Interpretiraj h'_j kot celo število i'_j za $1 \leq j \leq k$
  if $i'_1 \neq i'_2 \neq ... \neq i'_k$ **then**
    if $f(f(u_j)) = v_{i'_j}$ in $f(u_{j+\frac{k}{2}}) = v_{i'_{j+\frac{k}{2}}}$, $1 \leq j \leq \frac{k}{2}$ **then**
      Odgovor = `Sprejmi`
    end if
  end if
Izhod: Odgovor.

3 Analiza in primerjava

Poglavje pred nami nam poda samo računsko zahtevnost shem, predstavi nam velikost ključev in podpisa ter prikaže nivo varnosti.

3.1 Računska zahtevnost

Na računsko zahtevnost bomo gledali kot na število klicev enosmernih (zgoščevalnih) funkcij.

3.1.1 BiBa

Generiranje ključev:	t klicev funkcije f
Podpisovanje:	nastavljivo število klicev funkcije G_h in H
Verifikacija:	k klicev funkcije G_h in 1 klic funkcije H

Poglejmo sedaj kaj smo mislili z nastavljivim številom klicev funkcije G_h in H. Naj bo $P_k(x)$ verjetnost, da pride do natanko x k-kratnih trčenj. Velja torej $P_k(1) > P_k(2) > P_k(3)$. Za napadalca, ki poskuša ponarediti podpis in ima na voljo še manj "žog", $P_k(1) \gg P_k(2) \gg P_k(3)$. Podobno velja $P_k(1) \gg P_{k+1}(1)$ za podpisovalca in $P_{k+1}(1) = 0$ za napadalca.
Naj bo $E[k]$ pričakovano število košev, ki vsebujejo natanko k žog. Iz verjetnostnega računa je znano

$$E[k] = \frac{\binom{t}{k}(n-1)^{t-k}}{n^{t-1}}$$

Če pogledamo naslednji enačbi:

$$P_s \approx P_k(1) + P_k(2) + ...$$

$$E[k] = 1P_k(1) + 2P_k(2) + ...$$

vidimo, da je

$$P_s = E[k] - P_k(2) - 2P_k(3) - ...$$

kar pa je približno enako $E[k]$. Poskusimo najti še bolj preprosto obliko.

$$P_s \approx E[k] = \frac{\binom{t}{k}(n-1)^{t-k}}{n^{t-1}} < \frac{\binom{t}{k}(n)^{t-k}}{n^{t-1}} = \frac{\binom{t}{k}}{n^{k-1}} = \frac{t!}{(t-k)!k!n^{k-1}} < \frac{t^k}{k!n^{k-1}}$$

Velja torej:

$$P_s \approx \frac{t^k}{k!n^{k-1}}$$

Če želimo v povprečju 2 krat pognati algoritem, da dobimo uspešen podpis ($P_s = 50\%$) vidimo torej, da velja naslednja zveza: $n^{k-1} = \frac{2t^k}{k!}$.

3.1.2 HORS

Generiranje ključev:	t klicev funkcije f
Podpisovanje:	1 klic funkcije f
Verifikacija:	k klicev funkcije f in 1 klic funkcije H

3.1.3 Shema 1

Generiranje ključev:	$2t$ klicev funkcije f
Podpisovanje:	$\frac{t^k(\frac{k}{2}!)^2(t-k)!}{t!}$ klicev funkcije H v povprečju
Verifikacija:	$\frac{3k}{2}$ klicev funkcije f in 1 klic funkcije H

Poglejmo si sedaj podrobno zakaj imamo takšno povprečno število klicev funkcije H pri podpisovanju. Ker smo H modelirali v naključnem oraklu, ima ta enakomerno porazdelitev na intervalu $\left[0, 2^{k \log_2 t} - 1\right]$, kar pomeni, da imajo spremenljivke $i_1, ..., i_k$ enakomerno porazdelitev na $[1, t]$. Verjetnost, da velja $i_1 < ... < i_{k/2}$, je $\binom{t}{\frac{k}{2}}/t^{\frac{k}{2}}$. Ko smo $i_1, ..., i_{\frac{k}{2}}$ že izbrali, vidimo, da je verjetnost, da velja $i_{\frac{k}{2}+1} < ... < i_k$ in $\{i_1, ..., i_{\frac{k}{2}}\} \bigcap \{i_{\frac{k}{2}+1}, ..., i_k\} = \emptyset$, enaka $\binom{t-\frac{k}{2}}{\frac{k}{2}}/t^{\frac{k}{2}}$. Torej je verjetnost, da dobimo enačbi zadovoljiv podpis

$$\frac{\binom{t}{\frac{k}{2}}}{t^{\frac{k}{2}}} \frac{\binom{t-\frac{k}{2}}{\frac{k}{2}}}{t^{\frac{k}{2}}} = \frac{t!}{t^k(\frac{k}{2}!)^2(t-k)!}$$

Pričakovano število klicev funkcije H je torej temu obratna vrednost.
Če pogledamo časovno zahtevnost za $t = 1024$ in $k = 8$, vidimo, da mora podpisovalec v povprečju poklicati 592 klicev funkcije $H(m||c)$. Cena podpisovanja je torej kar velika.

3.1.4 Shema 2

Generiranje ključev:	$2t$ klicev funkcije f
Podpisovanje:	$\frac{t^k}{t(t-1)....(t-k+1)}$ klicev funkcije H v povprečju
Verifikacija:	$\frac{3k}{2}$ klicev funkcije f in 1 klic funkcije H

Če ponovno pogledamo število klicev funkije H za $t = 1024$ in $k = 8$, vidimo, da je povprečno število klicev enako 1.045. V povprečju imamo torej manj kot 2 klica funkcije $H(h||c)$.

3.2 Velikosti ključev in velikost podpisa

BiBa in HORS imata oba isto velikost ključev, tj. $t\ell$ bitov za javni ključ kot tudi za zasebni ključ, medtem ko Shema 1 in Shema 2 potrebujeta ravno $t\ell$ bitov za javni ključ, za zasebni ključ pa $2t\ell$ bitov. Seveda lahko $(p_1, ..., p_t)$ izračunamo iz $(s_1, ..., s_t)$.

Velikost podpisa za vse sheme je $k\ell$ bitov. Shema 1 in Shema 2 potrebujeta dodatnih $|c|$ bitov kar znese dodatnih 10 bitov pri Shemi 1 in 2 bita pri Shemi 2 za zgoraj navedene parametre $t = 1024$ in $k = 8$.

3.3 Varnostna analiza

V tem delu bomo analizirali varnost obravnavanih shem proti t.i. r-non-adaptive-message napadu. V tovrstnem napadu se za napadalca A predpostavi, da ima r sporočil, ki si jih sam izbere skupaj s pripadajočimi podpisi. Potem A poiskuša ponarediti podpis na novo izbranem sporočilu m'. Za poenostavitev analize bomo predpostavili, da A ne poiskuša poiskati inverza ali trčenja enosmerne fukcije f. Prav tako delamo analizo le za $r = 1$ glede na to, da so naše sheme za podpise enkratne.

3.3.1 BiBa

Naj bo P_f verjetnost, da napadalec ponaredi podpis. Velja, da je zgornja meja za P_f enaka

$$P_f = \frac{\binom{r}{k}(n-1)^{r-k}}{n^{r-1}}$$

kjer je r število komponent zasebnega ključa, ki jih napadalec pozna.

3.3.2 HORS

Ker ima napadalec le en podpis oz. k vrednosti izmed t, je očitno, da je verjetnost, da ponaredi podpis enaka $(\frac{k}{t})^k$.

3.3.3 Shema 1

Predpostavimo, da ima napadalec A veljaven podpis SIG $= (c, (s_{i_1}, ..., s_{i_{k/2}}), (p_{i_{k/2+1}}, ..., p_{i_k}))$ za sporočilo m. Veljaven podpis SIG$'$ za sporočilo m' lahko sestoji le iz elementov v SIG z izjemo c. Naj bo SIG$' = (c', S')$ tako da velja, da je S' sestavljen le iz elementov množice $\{s_{i_1}, ..., s_{i_{k/2}}, p_{i_{k/2+1}}, ..., p_{i_k}\}$. Med vsemi možnimi kandidati je le $S' = (s_{i_1}, ..., s_{i_{k/2}}), (p_{i_{k/2+1}}, ..., p_{i_k})$ tisti, ki bi bil uspešno verificiran. Torej mora veljati $H(m'||c') = i_1||...||i_k$. Ker ima H enakomerno porazdelitev velja

$$Pr\left[H(m'||c') = i_1||...||i_k\right] = \frac{1}{2^{|H()|}} = \frac{1}{2^{k\log_2 k}} = \frac{1}{t^k}$$

3.3.4 Shema 2

Podobno kot pri razmisleku pri Shemi 1 pridemo do ugotovitve, da je verjetnost, da napadalec ugotovi veljaven podpis enaka $\frac{((k/2)!)^2}{t^k}$.

3.4 Primerjava

V tabeli 1 imamo primerjavo obravnavanih shem pod pogojem, da imajo vse sheme enako velikost podpisa ($= k\ell$) in velikost javnega ključa ($= t\ell$). Pod tem pogojem je verjetnost ponaredbe P_f najmanjša pri Shemi 1 medtem ko je pri Shemi 2 manjša od tistih pri shemah HORS in BiBa za vse možne vrednosti (k, ℓ, t).

Za vse sheme velja, da se verjetnost ponaredbe podpisa poveča, če zmanjšamo vrednost parametra k. Če nastavimo k tako, da je verjetnost ponaredbe enaka pri vseh shemah, velja, da ima Shema 1 najmanjšo velikost podpisa, sledi pa ji Shema 2.

Cena generiranja ključev pri Shemi 1 in Shemi 2 je večja kot pri shemah HORS In BiBa. A glede na to, da je lahko generiranje ključev izvedeno vnaprej in na paralelen način z večimi strežniki, je ta pomankljivost zanemarljiva.

Shema	Cena verifikacije	Cena generiranja ključev	Cena podpisovanja	Verjetnost ponaredbe
BiBa	$2k+1$	t	$2t$	$\frac{k!}{2t^k}$
HORS	$k+1$	t	1	$\left(\frac{k}{t}\right)^k$
Shema 1	$\frac{3k}{2}+1$	$2t$	$\frac{t^k(\frac{k}{2}!)^2(t-k)!}{t!}$	$\left(\frac{1}{t}\right)^k$
Shema 2	$\frac{3k}{2}+1$	$2t$	$\frac{t^k}{t(t-1)....(t-k+1)}$	$\frac{((k/2)!)^2}{t^k}$

Tabela 1: Primerjava shem

4 Možna uporaba v praksi

Ker je navadno dobra praksa, da se najboljše pusti za na konec, je za sam konec predstavljena možna uporaba enkratnih digitalnih podpisov v praksi. Kako se v svetu računalniških komunikacij izognemo vsem varnostnim težavam pri oddajanju podatkov večji skupini prejemnikov nam predstavi

naslednje poglavje. Poglavje za njim pa nam prikaže kako s pomočjo istih konceptov pridemo do zadovoljivega nivoja varnosti v mobilnih ad-hoc omrežjih.

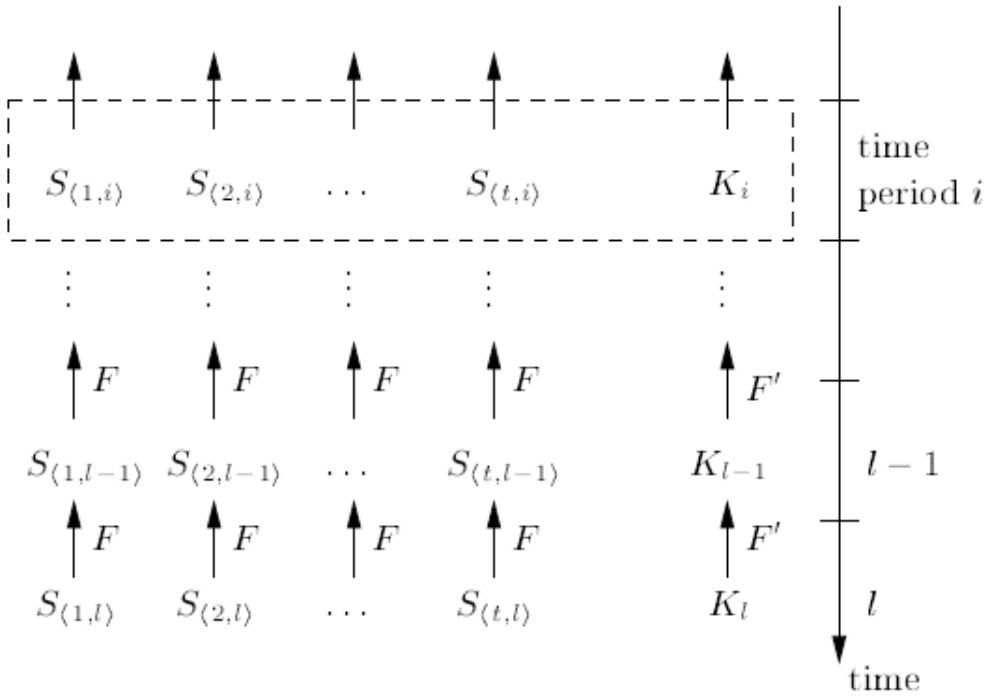
4.1 BiBa broadcast protokol z overjanjejm podatkov

Zadnja leta je eden izmed glavnih izzivov broadcast komunikacije, overjanje izvornih sporočil, ki prejemnikom omogoča preverjanje pristnosti izvora podatkov. Idealni broadcast protokol z overjanjem mora biti razširljiv in učinkovit za pošiljatelja ter prejemnika, imeti mora majhno režijo pri komunikaciji, prejemniku mora omogočiti preverjanje pristnosti vsakega sporočila posebej, biti mora odporen na izgubo paketov in nenazadnje pošiljatelju mora omogočati overiti neskončen tok sporočil oz. podatkov. V tem poglavju si bomo pogledali kako lahko uporabimo shemo BiBa za zgradbo broadcast protokola z overjanjem in dosežemo večino zgoraj omenjenih lastnosti.

V skladu s člankom, ki predstavlja shemo BiBa [4], poimenujmo vrednosti, ki predstavljajo skrivni ključ $(s_1, s_2, ..., s_t)$, SEAL[1] vrednosti. Prav tako uporabimo enosmerni fukciji $F : \{0,1\}^\ell \times \{0,1\}^m \rightarrow \{0,1\}^\ell$ in $F' : \{0,1\}^m \times \{0,1\}^m \rightarrow \{0,1\}^m$, s katerima zgradimo dve enosmerni verigi in dosežemo lastnost samodejnega overjanja in dodajanja SEAL vrednosti. Z uporabo funkcije F zgradimo t.i. *enosmerno SEAL verigo*, z F' pa t.i. *enosmerno salt verigo*.

Pošiljatelj najprej generira *enosmerno salt verigo* dolžine ℓ, $\{K_i\}_{1 \leq i \leq \ell}$, z uporabo funkcije F' na naslednji način: naključno najprej izbere K_ℓ (dolžine m bitov): $K_\ell \overset{R}{\leftarrow} \{0,1\}^m$ in nato rekurzivno izračuna vse druge *salt* vrednosti: $K_i = F_{K'_{i+1}}(0), (1 \leq i < \ell)$.
Nato generira množico *enosmernih SEAL verig*, $\{S_{\langle i,j \rangle}\}_{1 \leq i \leq t, 1 \leq j \leq \ell}$, kjer $S_{\langle i,\_ \rangle}$ tvori enosmerno verigo kot je prikazano na sliki 2. V prvem koraku najprej naključno izbere vse izvorne SEAL vrednosti $S_{\langle \_,\ell \rangle}$ dolžine ℓ bitov: $S_{\langle i,\ell \rangle} \overset{R}{\leftarrow} \{0,1\}^\ell (1 \leq i \leq t)$ nato pa rekurzivno izračuna $S_{\langle i,j \rangle} = F_{S_{\langle i,j+1 \rangle}}(K_{j+1}) (1 \leq j < \ell)$.

[1] angl. *SElfAuthenticating vaLues*.



Slika 2: Uporaba enosmernih verig

Pošiljatelj razdeli čas na časovne rezine dolžine T_d. V vsaki časovni rezini i so aktivne vrednosti SEAL $S_{\langle \_,i\rangle}$ in vrednost salt K_i. Ko preteče časovna rezina, veljavnost omenjenih SEAL vrednosti poteče, aktivne pa postanejo vrednosti $S_{\langle \_,i+1\rangle}$. Pošiljatelj objavi vsako salt vrednost v začetku časovne rezine, medtem, ko izmed SEAL vrednosti, postanejo na voljo le tiste, ki tvorijo podpis.

Da se pridruži nov prejemnik lahko zaenkrat predpostavimo, da pošiljatelj prejemniku pošlje vse SEAL in salt vrednosti prejšnje časovne rezine po varnem kanalu. Prejemnik preveri pristnost podatkov tako da preveri $K_i \stackrel{?}{=} F'_{K_{i+1}}(0)$ nato pa še preveri pristnost SEAL vrednosti z uporabo *enosmernih SEAL verig* nazaj do SEAL vrednosti za katere ve, da so pristne.

Varnostne zahteve narekujejo, da morata biti pošiljatelj in prejemnik časovno sinhornizirana. Le tako lahko namreč prejemnik preveri ali napadalec zagotovo pozna le majhno število SEAL vrednosti. Recimo, da je maksimalna napaka časovne sinhornizacije δ. Pošiljatelj lahko v času δ

pošlje kvečjemu $\lfloor r/k \rfloor$ sporočil, kjer r predstavlja maksimalno število aktivnih SEAL vrednosti, ki jih napadalec lahko pozna, k pa število SEAL vrednosti, ki sestavljajo podpis. Ker pošiljatelj ne more več uporabiti ene BiBa instance za tem, ko je razrkil že r SEAL vrednosti, se uporabi več BiBa instanc in s tem doseže zvezno in neprekinjeno pošiljanje.

Predstavljeni protokol ima pri komunikaciji nizko režijo, popolno robustnost izgube paketov, pa vendar gre še vedno kar nekaj računske režije za preverjanje pristnosti na strani prejemnika. V sami predstavitvi sheme BiBa [4] oz. broadcast protokola z overjanjem, si lahko v poglavju 4 pogledate razširitvi protokola tako, da imamo namesto zgoraj omenjene pomankljivosti, bodisi neodpornost na izgubo paketov bodisi večjo režijo pri komunikaciji.

4.2 Varno AODV usmerjanje

Mobilna ad-hoc omrežja (MANET[2]) so specifična omrežja, ki jih sestavljajo mobilne naprave kot so prenosni računalniki, PDA naprave, mobilni telefoni itd. Ker tovrstna omrežja ne slonijo na klasični infrastrukturi, mora vsako vozlišče oz. naprava sodelovati pri usmerjanju in posredovanju podatkov ostalim vozliščem. Kljub temu, da obstaja kar nekaj MANET usmerjevalnih protokolov, jih večina nima varnostnih mehanizmov. V kombinaciji z omejitvami računskih virov in pasovne širine, predstavlja implementacija varnosti v MANET omrežjih težavo vredno izziva.

Pogledali si bomo AODV[3] usmerjevalni protokol z overjanjem s katerim odpravimo napade kot so nepooblaščeno sodelovanje, lažno usmerjanje, spreminjanje usmerjevalnih sporočil itd. Glavni gradnik protokola je shema za enkratne podpise HORS, ki smo jo spoznali v podpoglavju 3.1.2. Za začetek omenimo, da AODV spada pod reaktivne protokole pri katerih se pot poišče po potrebi s poplavljanjem zahtev po informaciji o poti do ciljnega omrežja oz. vozlišča. To se doseže z uporabo naslednjih tipov paketov: PREQ (route request), RREP (route reply) in RRER (route error). Prav tako se uporablja sekvenčne številke za preprečevanje usmerjevalnih zank.

[2]Mobile Ad-hoc NETworks

[3]Ad-hoc On demand Distance Vector

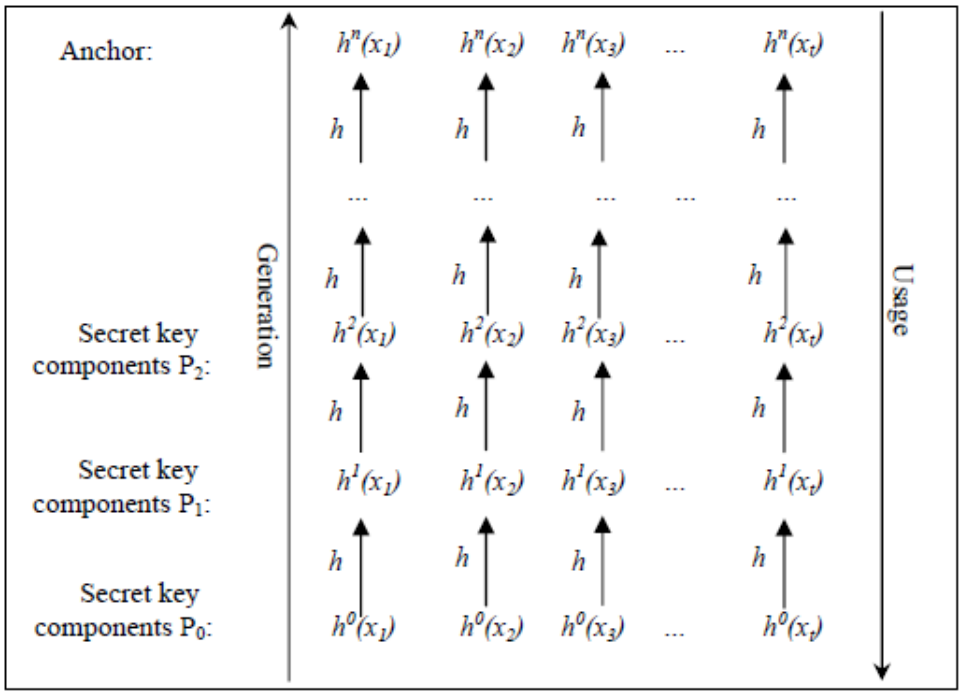
4.2.1 Generiranje ključev

Uporabimo notacijo:

$h(), h, h^i()$	- enosmerne zgoščevalne funkcije;
Sign_{K_n}	- klasični digitalni podpis (RSA) generiran od vozlišča n;
$\langle m \rangle K_n^{-1}$	- enkratni digitalni podpis vozlišča n nad sporočilom m.

Generiranje verige ključev:

1. Vsako vozlišče izbere t naključnih vrednosti x_j $(j = 1, ..., t)$.
2. Vsako vozlišče ustvari n verig dolžine t (glej sliko 3).
3. Komponente javnega ključa dobimo z uporabo enosmerne zgoščevalne funkcije h.
4. Javni ključ razkrijemo vsakič, ko se zamenja.



Slika 3: Veriga komponent zasebnega ključa

4.2.2 AODV usmerjevalni protokol z overjanjem

Protokol [7], ki ga bomo spoznali, bo vseboval naslednje varnostne lastnosti:

1. Ciljno vozlišče lahko overi izvorno vozlišče.
2. Vsako vozlišče B, ki neposredno prejme usmerjevalni paket od vozlišča A, lahko preveri pristnost, da je paket resnično prišel od vozlišča A.
3. Vsako vmesno vozlišče lahko overi pošiljatelja in si s tem omogoči posodobitev usmerjevalne tabele.
4. *Hop count* vrednost je zaščitena z uporabo enosmernih verig.

Klasični digitalni podpis[4] bo uporabljen za overjanje paketov na izvornem vozlišču medtem, ko bo enkratni podpis namenjen za overjanje in preverjanje pristnosti na relaciji sosednjih vozlišč.

Enkratni javni ključ je s strani poljubnega vozlišča razdeljen vsem sosednjim vozliščem. Da dosežemo, da ima vsako vozlišče pravilen javni ključ, prvi javni ključ sploh, t.i. *anchor*, varno razdelimo na način, ki bo opisan v naslednjem odstavku. Zaporedne javne ključe lahko učinkovito razdelimo z paketi tipa *Hello*[5], ki se pošiljajo periodično. Preverjanje pristnosti novih ključev je enostavno - prvi zasebni ključ ZK_1 je $(k, h^n(x_1), h^n(x_2), h^n(x_3), ..., h^n(x_t))$, pripadajoči javni ključ JK_1 pa $(k, h^{n+1}(x_1), h^{n+1}(x_2), h^{n+1}(x_3), ..., h^{n+1}(x_t)$. Drugi zasebni ključ ZK_2 je enak $(k, h^{n-1}(x_1), h^{n-1}(x_2), h^{n-1}(x_3), ..., h^{n-1}(x_t))$ in JK_2 enak $(k, h^n(x_1), h^n(x_2), h^n(x_3), ..., h^n(x_t))$ kar se lahko preveri s preslikavo h in primerjanjem z JK_1.

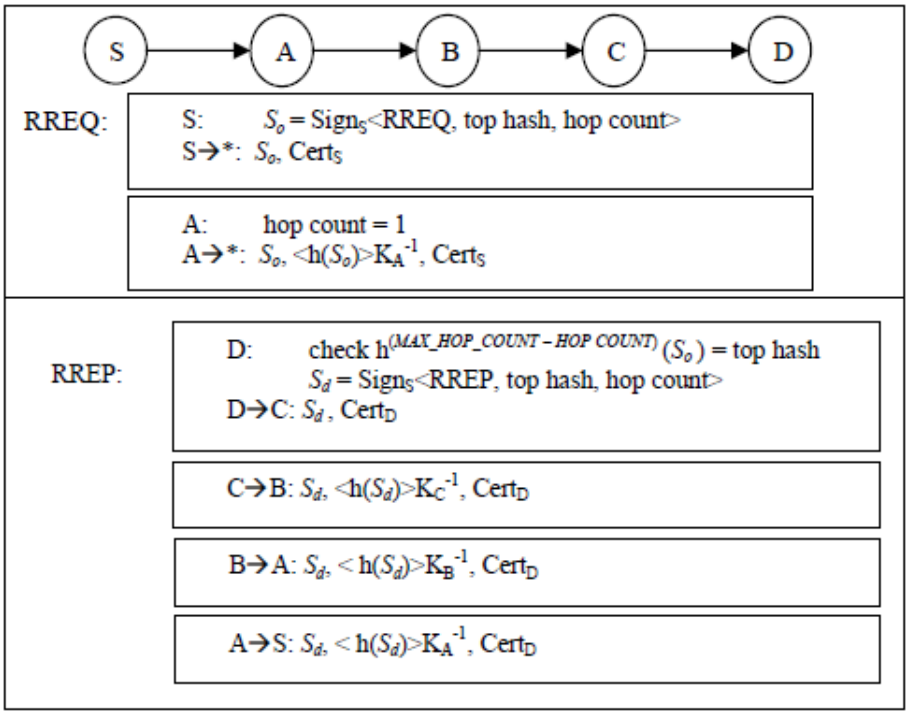
Poglejmo sedaj kaj se zgodi, ko se v omrežje pridruži novo vozlišče n. Najprej naključno izbere komponente zasebnega ključa in zgradi enosmerno zgoščevalno verigo nato pa podpiše prvi javni ključ (*anchor*) s klasičnim podpisom: $\mathrm{Sign}_{K_n}\langle N, \mathrm{JK}_1, \mathrm{Timestamp}\rangle, \mathrm{Cert}_N$.

Sedaj si bomo pogledali kako se poišče pot oz. kako se izvede t.i. *Route Discovery* (slika 4). Izorno vozlišče, ki da zahtevo za *Route Discovery*, samo

[4]Predpostavimo obstoj izdajatelja digitalnih potrdil, ki vozlišču le to izda pred vključitvijo v omrežje.

[5]S paketi tega tipa se v omrežju tvorijo relacije sosednjih vozlišč.

zahtevo podpiše s klasičnim podpisom. Neko prvo vozlišče A prejme zahtevo in naprej preveri pravilnost podpisa in v kolikor je verifikacija pravilna, z h preslika prejeti paket S_0 in ga podpiše s shemo za enkratni podpis. Ko drugo vozlišče prejme dvojno podpisan paket RREQ, preveri najprej pravilnost enkratnega podpisa in ponovno ob pravilnem podpisu, najprej s h preslika sporočilo S_0 nato pa rezultat podpiše s shemo HORS in paket odda naslednjim sosednjim vozliščem. Vidimo, da lahko neko vozlišče posodobi svojo tabelo le, če sta klasični in enkratni podpis pravilna.
Ko RREQ prispe do ciljnega vozlišča D, le to preveri pristnost paketa na isti način kot vmesna vozlišča, generira in podpiše RREP paket na isti način kot se je generiral RREQ in pošlje paket nazaj direktno do vozlišča, ki je izdal zahtevo.



Slika 4: Usmerjevalni zahtevek in odgovor

Literatura

[1] Yookun Cho. Efficient one-time signature schemes for stream authentication. *Journal of Information Science and Engineering*, 22:611–624, 2006.

[2] Aleksandar Jurišić. Kriptografija in teorija kodiranja. `http://lkrv.fri.uni-lj.si/~ajurisic/kitk2-09/folije/p09.pdf`, 2009.

[3] Aleksandar Jurišić. Zaupati ali ne zaupati – digitalni podpis v kriptografiji, 4. del. `http://lkrv.fri.uni-lj.si/popularizacija/presek/Jurisic_Zaupati_ali_ne_zaupati_Digitalni_podpis_v_kriptografiji_4.pdf`, 2011.

[4] Adrian Perrig. The biba one-time signature and broadcast authentication protocol. In *Computer and Communications Security*, pages 28–37, 2001.

[5] Adrian Perrig and Michael Mitzenmacher. Bounds and improvements for biba signature schemes.

[6] Leonid Reyzin and Natan Reyzin. Better than biba: Short one-time signatures with fast signing and verifying. In *Australasian Conference on Information Security and Privacy*, pages 144–153, 2002.

[7] Shidi Xu, Yi Mu, and Willy Susilo. Secure aodv routing protocol using one-time signature. In *Mobile Sensor Networks*, pages 288–297, 2005.