Kryptografické protokoly

použijeme-li při tvorbě systému k řešení nějakého problému odpovídající protokol, stačí pouze ověřit korektnost implementace vzhledem k tomuto protokolu

1. *arbitrované protokoly* - *arbitrem* rozumíme neutrální třetí stranu, zajišťující férovost, nevýhodou použití arbitra jsou zvýšené náklady na provoz, problémy s dostupností, časové prodlevy, potíže s důvěryhodností a výkonem

2. *rozhodované (adjudicated) protokoly* - rozhodčí (adjudicator) je třetí strana, která je schopna rozhodnout, zda operace byla vykonána férově, případně kdo z účastníků porušuje pravidla, třetí strana je tedy používána pouze při sporu

3. *samozabezpečovací (self-enforcing) protokoly* - samotný protokol zajišťuje vzájemnou ochranu účastníků

Digitální podpisy

musí být nefalšovatelné, autentické, neměnitelné, "nerecyklovatelné", ověřitelné

Protokol pro symetrické systémy

Nechť odesílatel *S* zasílá příjemci *R* zprávu *M*.
1. *S* zašle arbitrovi *A* zprávu $E(M, K_S)$
2. Arbiter verifikuje odesílatele a příjemci *R* zašle $E((M, S, E(M, K_S)), K_R)$
3. Příjemce uschová *M* a $E(M, K_S)$ pro účely případného dokazování přijetí

v případě, že nepožadujeme utajení přenášených dat, vystačíme s použitím MAC namísto šifrování

na obranu proti opakovanému použití došlé zprávy lze použít vhodnou časovou známku, proti sestavování nových zpráv z částí dříve došlých poslouží časová známka v každém šifrovaném bloku.

Protokol pro asymetrické systémy

velice jednoduché, provede se

$$D(M, K_S)$$

dnes vzhledem k výkonu asymetrických šifer probíhá výpočet:

$$Sign = \{\langle M \rangle\}_{K_S}^{-1}$$

takto vzniká tzv. oddělený (detached) podpis, obvykle je tento podpis připojen k vlastním podepisovaný datům
ověření podpisu:

$$\langle \quad \rangle \left\{ \langle \quad \rangle \right\}$$

Poker

Nechť *A* hraje s *B*, *A* rozdává

1. *A* vezme karty, náhodně je seřadí a zašifruje svým klíčem, výsledek zašle *B*
2. *B* vybere *m* balíčků, zašifruje je svým klíčem a společně s dalšími *m* balíčky zašle zpět.
3. *A* dešifruje všechny zašifrované balíčky. *M* z nich je dosud zašifrováno K_B , ty zašle zpět *B*.
4. *B* dešifruje přijaté balíčky
5. oba hráči mají k dispozici své karty

protokol je snadno rozšiřitelný pro větší počet hráčů, důkaz korektnosti hry se provádí zveřejněním utajovaných skutečností tak, aby každý z hráčů mohl provést opakování celé hry

pro úspěšné zvládnutí kroku 3. je nutné, aby šifra komutovala

popsanou implementaci pomocí symetrických systémů lze velmi snadno převést na systémy asymetrické

protokol má praktické využití:

distribuce šifrovacích klíčů

- někdy je žádoucí, aby ani klíč-server neznal uživatelův tajný klíč. Server tedy vytvoří množství balíčků s klíči, žadatel si jeden vybere a postupem shodným s předchozím protokolem získá ukrytý klíč.

Časové známky

nejjednodušší metodou je zasílat kopie zpráv důvěryhodnému arbitrovi, problémy s množstvím uchovávaných dat lze vyřešit použitím hašovacích funkcí

Spojované (linked) známky

- aby odesílatel (adresát) společně s arbitrem nemohli podvádět

1. Odesílatel *S* zašle arbitrovi *A* hashkód zprávy H_n.

2. *A* vrátí odesílateli

$$T_n = S_K\left(n, S, H_n, Tm_n; ID_{n-1}, H_{n-1}, T_{n-1}, \mathbf{H}\left(ID_{n-1}, H_{n-1}, T_{n-1}\right)\right)$$

   kde n je pořadí zprávy, Tm_n čas podpisu zprávy, Id_{n-1} … jsou informace o předešlé zprávě, kterou arbitr vyřizoval
3. po vyřízení následující zprávy arbitr zašle odesílateli identifikaci následujícího odesílatele

Chce-li někdo ověřit časovou známku zprávy, kontaktuje odesílatele Id_{n-1} a Id_{n+1} a pomocí nich ověřit platnost T_n.

Pro zvýšení bezpečnosti je možné připojit informace o více předchozích zprávách a držet seznam stejného počtu následujících odesílatelů

Další možností je tento protokol:

1. H_n užijeme jako vstup vhodného generátoru náhodných čísel, m nejbližších výsledků bereme jako identifikace uživatelů $V_1, \ldots, V_m$
2. všem V_i zašle S hashkód H_n
3. V_i připojí informaci o času, celý balíček elektronicky podepíše a vrátí S
4. S uschová všechny odpovědi jakožto časovou známku

použití generátoru náhodných čísel v 1. kroku zajišťuje, že S nemůže podvádět, neb neví předem, kdo bude jeho zprávu podepisovat, navíc okruh verifikátorů se bude pokaždé měnit

Fixace bitu (Bit commitment)

občas je potřeba mít možnost v budoucnu protistraně dokázat znalost jisté informace bez jejího předčasného vyzrazení
nechť strana *A* dokazuje straně *B* znalost skutečnosti *b*

1. *B* zašle entitě *A* náhodný řetězec *R*
2. *A* spojí *R* se svojí informací celou tuto zprávu zašifruje náhodný klíčem *K* a výsledek zašle *B*

$$E\left(K, R \| b\right)$$

3. při pozdějším ověřování *A* zašle *B* klíč *K*, *B* dešifruje přijatou zprávu, ověří přítomnost *R* a přezkoumá *b*

s použitím one-way funkcí se můžeme omezit pouze na jednosměrnou komunikaci
1. *A* vygeneruje dva náhodné řetězce R_1 a R_2
2. *A* vytvoří zprávu obsahující R_1 , R_2 a *b,* spočítá hashkód této zprávy a *B* zašle

$$\mathbf{H}\left(R_1 \| R_2 \| b\right), R_1$$

3. v rámci dokazování znalosti *b* zašle *A* původní zprávu

$$\left(R_1 \| R_2 \| b\right)$$

4. *B* spočítá hashkód a porovná R_1

Důkazy s nulovou informací (zero-knowledge proofs)

- dokazovateli (prover) nesmí být umožněno podvádět, pokud důkaz nezná, jeho šance přesvědčit ověřovatele je mizivá
- ověřovatel rovněž nesmí podvádět, o samotném důkazu smí zjistit pouze to, že jej dokazovatel zná. Zvláště nesmí být schopen celý důkaz rekonstruovat a sám provést.
- ověřovatel se nesmí dozvědět nic, co by nebyl schopen zjistit bez pomoci dokazovatele.

není-li splněna poslední podmínka, mluvíme o *důkazech s minimálním vyzrazením* (minimum-disclosure proofs)

jeden z možných důkazů založen na problematice Hamiltonovských kružnic v grafu
1. Nechť *A* zná Hamiltonovskou kružnici v grafu *G*
2. *A* provede náhodnou permutaci *G.* Původní graf a vzniklý *H* jsou izomorfní.
3. Kopie grafu *H* je zaslán entitě *B*
4. Ověřovatel *B* položí dokazovateli *A* jednu z následujících otázek:
   a) dokázat, že *G* a *H* jsou izomorfní
   b) ukázat Hamiltonovskou kružnici v grafu *H*
5. opakováním kroků 1. až 4. lze docílit potřebné jistoty

Neurčitý přenos (Oblivious transfer)

protokol umožňuje, aby si adresát vybral z několika nabízených možností aniž by odesílatel předem znal jeho volbu, možné doplnění o následnou vzájemnou kontrolu

A vygeneruje dva páry public-key klíčů, oba veřejné klíče zašle *B*

B vytvoří klíč *K* pro symetrický algoritmus, tento klíč zašifruje jedním z přijatých klíčů a výsledek vrátí *A*
A dešifruje přijatou zprávu oběma tajnými klíči, čímž získá K_1 a K_2
Klíčem K_1 zašifruje *A* jednu z posílaných zpráv, klíčem K_2 druhou a oba výsledky zašle *B*.
B se pokusí dešifrovat přijaté zprávy, přičemž v jednom případě získá smysluplný výsledek
Případné ověření se provede tak, že *A* zveřejní své tajné klíče.

Protokol sám o sobě lze používat k distribuci šifrovacích klíčů, případně jako obdobu házení korunou. Větší význam má jako součást následujícího protokolu.

Podepisování kontraktů (Contract signing)

V každém okamžiku musí být obě smluvní strany vázány stejně moc

nejjednodušším řešením je arbitrovaný protokol, kde obě strany předají centrální autoritě své podepsané kopie a tato třetí strana zajistí výměnu po obdržení obou kopií

daleko lepší je následující distribuovaný protokol:

1. *A* i *B* náhodně vygenerují 200 konvenčních klíčů, které rozdělí do dvouprvkových množin
2. *A* i *B* vytvoří 100 párů zpráv L_n a R_n, zhruba ve tvaru "Toto je levá část *n*-tého podpisu smlouvy "…, každá ze zpráv navíc obsahuje polovinu elektronického popisu smlouvy, timestamp atd. Kontrakt považujeme za podepsaný druhou stranou, pokud se můžeme prokázat oběma polovinami některého z podpisů.
3. *A* i *B* zašifrují *i*-tý pár zpráv *i*-tým párem klíčů ("levou" zprávu jedním, "pravou" druhým z klíčů)
4. obě strany si navzájem zašlou páry zašifrovaných zpráv (200 zpráv každý)
5. použitím protokolu pro Oblivious transfer si *A* a *B* navzájem zašlou všechny šifrovací klíče - druhá strana má tedy z každého páru jeden (který ?) klíč
6. *A* i *B* provedou dešifrování těch zpráv, ke kterým mají k dispozici odpovídající klíč
7. *A* zašle *B* první bit všech 200 konvenčních klíčů, obdobně *B*
8. předchozí krok opakujeme dokud nejsou přeneseny všechny bity klíčů
9. obě smluvní strany mohou dešifrovat všechny zprávy -> kontrakt je podepsán

Pokud by se některá ze stran pokusila o podvod v kroku 4. nebo 5., bude podvod odhalen v kroku 6. Podvod v kroku 7. bude s velkou pravděpodobností objeven okamžitě. Ukončí-li jeden z účastníků protokol před zasláním všech bitů klíčů, mají oba stejnou šanci dopočítat některý z klíčů a získat elektronický podpis druhého. Problémem je, má-li někdo z podepisujících výrazně větší výpočetní kapacitu, v protokolu není zlom, ve kterém by se výrazně změnila míra vázanosti účastníků.

Elektronická potvrzovaná pošta (digital certified mail)

chceme, aby adresát mohl přečíst naši zprávu až poté, co získáme potvrzení o tom, že ji obdržel (elektronický doporučený dopis)

1. *A* zašifruje posílanou zprávu náhodně zvoleným konvenčním klíčem *K*
2. *A* vytvoří 100 párů konvenčních klíčů - první klíč každého páru je generován náhodně, druhý je XOR prvního klíče a klíče *K*
3. *A* zašifruje pomocnou zprávu každým ze 100 párů těchto klíčů (200 šifer)
4. všechny výsledné páry šifer zašle *B*
5. *B* vygeneruje 100 párů náhodných konvenčních klíčů
6. *B* vytvoří 100 párů zpráv tvaru "Toto je levá část mého potvrzení číslo *n*". Opět potvrzení o přjetí je platné, pokud se protější strana může prokázat oběma polovinami jednoho z exemplářů potvrzení.
7. *B* zašifruje *i*-tý pár zpráv *i*-tým párem klíčů ("levou" zprávu jedním, "pravou" druhým z klíčů)
8. výsledné páry šifer zašle *B* protější straně
9. s použitím protokolu pro Oblivious transfer si *A* i *B* navzájem pošlou všech 100 párů svých klíčů - žádná ze stran neví, který klíč z kterého páru protějšek má k dispozici
10. *A* i *B* dešifrují všechny zprávy, ke kterým mají klíče a ověří jejich smysluplnost
11. obě strany si zašlou první bit všech 200 svých šifrovacích klíčů
12. předchozí krok je opakován dokud nejsou přeneseny všechny bity
13. *A* i *B* dešifrují zbývající části párů zpráv, které v předchozích krocích obdrželi - *A* má k dispozici potvrzení o přijetí zprávy od *B* a *B* může provést XOR libovolného páru klíčů a dešifrovat zprávu.

Použití pomocné zprávy dává straně *B* možnost odhalení podvodu v kroku 10.
V ostatních ohledech má protokol obdobné vlastnosti jako protokol pro podepisování kontraktů.

Bezpečné volby

- volit smí pouze oprávnění voliči
- každý smí hlasovat nejvýše jednou
- nikdo nesmí vědět, kdo jak volil
- nikdo nesmí měnit volbu jiných
- každý hlas musí být započítán
- je zjistitelné, kdo volil

Poslední podmínka již není nutná.

Protokol se dvěma centrálními autoritami

používá registrační autoritu *RA* provádějící registraci voličů a sčítací autority *SA*, která sčítá hlasovací lístky a zveřejňuje výsledky voleb

1. všichni voliči zašlou *RA* žádost o validační číslo
2. *RA* zašle každému voliči náhodně zvolené validační číslo *L*, zároveň si ponechá seznam, kdo jaké číslo dostal
3. *RA* zašle seznam validačních čísel *SA*
4. každý z voličů si náhodně vybere svoje identifikační číslo *Id* a *SA* zašle zprávu

$$(L, Id, v)$$

   kde *v* je volba
5. *SA* porovná *L* se seznamem validačních čísel z kroku 3. odpovídající číslo škrtne a voličovo *Id* přidá do seznamu asociovaného s voleným kandidátem
6. po skončení voleb *SA* zveřejní výsledky voleb a seznamy identifikačních čísel spojené se jmény kandidátů

Každý z voličů si může ověřit zda byl jeho hlas správně započítán, *RA* zjistí případné falešné hlasy.
SA však může registrovat neoprávněné voliče, případně některé voliče vícekrát. V případě že budou *RA* a *SA* spolupracovat, hrozí nebezpečí porušení anonymity voličů.

Protokol bez centrální autority

voliči *A, B, C, D* se rozhodují ano či ne

1. všichni voliči zváží své rozhodnutí a
   (a) ke své volbě připojí náhodný řetězec
   (b) zašifrují výsledek kroku (a) veřejným klíčem voliče *D*
   (c) zašifrují výsledek kroku (b) veřejným klíčem voliče C
   (d) zašifrují výsledek kroku (c) veřejným klíčem voliče *B*

(e) zašifrují výsledek kroku (d) veřejným klíčem voliče *A*
(f) připojí k výsledku kroku (e) nový náhodný řetězec, jehož hodnotu uchovají, a celé to zašifrují veřejným klíčem voliče *D*
(g) připojí k výsledku kroku (f) nový náhodný řetězec, jehož hodnotu uchovají, a celou zprávu zašifrují veřejným klíčem voliče *C*.
(h) připojí k výsledku kroku (g) nový náhodný řetězec, jehož hodnotu uchovají, a celou zprávu zašifrují veřejným klíčem voliče *B*.
(i) připojí k výsledku kroku (h) nový náhodný řetězec, jehož hodnotu uchovají, a celou zprávu zašifrují veřejným klíčem voliče *A*.

vcelku tedy

$$E_A\Big(R_5, E_B\Big(R_4, E_C\Big(R_3, E_D\Big(R_2, E_A\Big(E_B\Big(E_C\Big(E_D\big(v, R_1\big)\Big)\Big)\Big)\Big)\Big)\Big)\Big)$$

kde R_i je náhodný řetězec a v voličovo rozhodnutí

2. všichni voliči pošlou výsledky svých výpočtů voliči *A*
3. *A* dešifruje všechny přijaté zprávy svým tajným klíčem, otestuje přítomnost svého hlasu a ze všech zpráv oddělí R_5 .
4. *A* zamíchá pořadím zpráv a všechny pošle voliči *B*. Zprávy mají tvar

$$E_B\Big(R_4, E_C\Big(R_3, E_D\Big(R_2, E_A\Big(E_B\Big(E_C\Big(E_D\big(v, R_1\big)\Big)\Big)\Big)\Big)\Big)\Big)$$

5. *B* dešifruje všechny přijaté zprávy svým tajným klíčem, otestuje přítomnost svého hlasu a ze všech zpráv oddělí R_4 .
6. *B* zamíchá pořadím zpráv a všechny pošle voliči *C*. Zprávy mají tvar

$$E_C\Big(R_3, E_D\Big(R_2, E_A\Big(E_B\Big(E_C\Big(E_D\big(v, R_1\big)\Big)\Big)\Big)\Big)\Big)$$

C dešifruje všechny přijaté zprávy svým tajným klíčem, otestuje přítomnost svého hlasu a ze všech zpráv oddělí R_3 .
7. *C* zamíchá pořadím zpráv a všechny pošle voliči *D*. Zprávy mají tvar

$$E_D\Big(R_2, E_A\Big(E_B\Big(E_C\Big(E_D\big(v, R_1\big)\Big)\Big)\Big)\Big)$$

D dešifruje všechny přijaté zprávy svým tajným klíčem, otestuje přítomnost svého hlasu a ze všech zpráv oddělí R_2 .

8. *D* zamíchá pořadím zpráv a všechny pošle voliči *A*. Zprávy mají tvar

$$E_A\left(E_B\left(E_C\left(E_D\left(v, R_1\right)\right)\right)\right)$$

A dešifruje všechny přijaté zprávy svým tajným klíčem, ověří přítomnost svého hlasu, všechny zprávy elektronicky podepíše a zašle všem ostatním.

$$D_A\left(E_B\left(E_C\left(E_D\left(v, R_1\right)\right)\right)\right)$$

9. *B* ověří a smaže elektronický podpis voliče *A*, dešifruje všechny přijaté zprávy svým tajným klíčem, ověří přítomnost svého hlasu, všechny zprávy elektronicky podepíše a zašle všem ostatním.

$$D_B\left(E_C\left(E_D\left(v, R_1\right)\right)\right)$$

10. *C* ověří a smaže elektronický podpis voliče *B*, dešifruje všechny přijaté zprávy svým tajným klíčem, ověří přítomnost svého hlasu, všechny zprávy elektronicky podepíše a zašle všem ostatním.

$$D_C\left(E_D\left(v, R_1\right)\right)$$

11. *D* ověří a smaže elektronický podpis voliče *C*, dešifruje všechny přijaté zprávy svým tajným klíčem, ověří přítomnost svého hlasu, všechny zprávy elektronicky podepíše a zašle všem ostatním.

$$D_D\left(v, R_1\right)$$

12. Všichni verifikují a odstraní elektronický podpis voliče *D* a ověří, že jejich hlas je stále přítomen.
13. Každý může sám spočítat výsledky voleb.

Protokol zabraňuje tomu, aby libovolná ze zúčastněných stran neoprávněně manipulovala s rozhodnutíni ostatních voličů, rovněž anonymita všech voličů je zajištěna.
Slabinou protokolu je značná komplikovanost, navíc volič *D* má výsledky voleb k dispozici dřív než ostatní.

Bezpečné spolupočítání (secure multiparty computation)

Protokol umožňuje skupině uživatelů počítat funkci nad vstupními daty tak, aby všichni znali výsledek ale ne vstup ostatních.

Počítání průměrné hodnoty

1. *A* přičte ke své hodnotě tajné náhodné číslo, výsledek zašifruje veřejným klíčem účastníka *B* a předá výsledek *B*.
2. *B* dešifruje zprávu, přičte svoji hodnotu, zašifruje a pošle dalšímu účastníku.
3. Poslední z účastníků dešifruje přijatou zprávu, přidá svoji hodnotu, výsledek zašle *A*.
4. *A* po dešifrování odečte své náhodné číslo, spočítá výsledek a zveřejní jej.

Protokol nezabrání účastníku *A* podvádět, ani nezajišťuje, aby ostatní zadali správné hodnoty.

Porovnávání čísel

Nechť *A* má tajné číslo *i* a *B* má tajné číslo *j*. Nechť *i* a *j* jsou celá čísla mezi 1 a 100

1. *A* náhodně zvolí velké číslo *x* zašifuje je veřejným klíčem entity *B* a zašle:
$$c - i = \mathrm{E}(K_B, x) - i$$
2. *B* spočítá těchto 100 čísel:
$$y_u = \mathbf{D}(K_B, c - i + u)\ ,\ 1 \le u \le 100$$
vybere náhodně prvočíslo *p* o málo menší než *x* a spočítá všechna
$$z_u = (y_u \bmod p)\ ,\ 1 \le u \le 100$$
3. *B* ověří, že pro všechna $u \neq v$:
$$|z_u - z_v| \ge 2$$
a pro všechna *u*
$$0 < z_u < p - 1$$
V případě neúspěchu opakuje předchozí bod.
4. *B* zašle *A* následující sekvenci:
$$z_1, z_2, \ldots, z_j, z_{j+1} + 1, z_{j+2} + 1, \ldots, z_{100} + 1, p$$
5. *A* zjistí, zda *i*-tý prvek posloupnosti je kongruentní s *x* mod *p*. Tehdy a jen tehdy je $i \le j$. *A* oznámí výsledek.

Vadou protokolu je fakt, že v posledním kroku může *A* zastavit výpočet, nebo podvádět. Řešením paralelní provádění protokolu oběma stranami v kombinaci s vhodným protokolem pro výměnu zpráv.

Podpisy naslepo (blind signatures)

občas potřebujem ověřit dokument, aniž by ověřující znal jeho obsah

Možným řešením je následující protokol. Nechť *B* má šifrovací klíč *e*, dešifrovací *d*. S *A* sdílejí modul *n*. *A* chce od *B* ověřit dokument *m.*

1. *A* náhodně zvolí *k* mezi 1 a *n*. Entitě *B* zašle:
$$t \equiv m(k)^e \pmod{n}$$
2. *B* přijatou zprávu podepíše a zašle zpět:
$$t^d \equiv \left(m(k)^e\right)^{\mathrm{d}} \pmod{n}$$
3. *A* odkryje původní zprávu:
$$s \equiv t^d / k \pmod{n} \equiv m^d \pmod{n}$$

číslu *k* říkáme *oslepující faktor* (blind faktor).

Protokol lze rozšířit tak, aby ověřující neznal pouze část zprávy. V kroku 1. posíláme *i* zpráv, které obsahují proměnné pole. V kroku 2. navíc *B* požádá o oslepující faktor *i* - 1 náhodně zvolených zpráv, které si tak může prohlédnout celé. Podepíše zbývající zprávu a vrátí ji. *B* tedy nezná obsah proměnného pole podepsané zprávy.

Elektronické platby (digital cash)

problém kreditních karet spočívá v sledovatelnosti toku peněz. Hledáme protokol pro tvorbu autentizovaných ale nesledovatelných zpráv.

1. Zákazník *A* připraví 100 anonymních příkazů k platbě na stejnou částku. Každý z příkazů vypadá následovně:

Množství:	Kč 1,-
Jednozn. řetězec:	*X*

Identifikační řetězce:

$$\left(I_{1_L}, I_{1_R}\right)$$
$$I_2 = \left(I_{2_L}, I_{2_R}\right)$$
$$\dots$$
$$I_{100} = \left(I_{100_L}, I_{100_R}\right)$$

Každý příkaz obsahuje 100 různých párů identifikačních řetězců, každý vzniklý z řetězce obsahujícího úplnou identifikaci zákazníka vhodným algoritmem pro secrets splitting. Např. každý pár může být tvořen párem paketů podobně jako v protokolu pro bit commitment tak, aby se dalo kontrolovat rozkrytí paketu. Známe-li obě poloviny, můžeme sestavit původní řetězec.

2. *A* "zaslepí" všechny příkazy protokolem pro podpisy naslepo a odešle je do banky *B*.
3. *B* požádá *A* o "odslepení" náhodně zvolených 99 příkazů a rozkrytí všech identifikačních řetězců
4. Je-li vše v pořádku, banka *B* podpisem potvrdí zbylý příkaz a vrátí jej *A*.
5. *A* "odslepí" potvrzený příkaz a předá jej obchodníkovi.
6. Obchodník ověří podpis banky a tím legitimnost příkazu.
7. Obchodník požádá *A* náhodně o rozkrytí jedné poloviny každého páru identifikačních řetězců.
   +
8. *A* splní požadavek.
9. *B* ověří svůj podpis a zjistí, zda již nepřijala příkaz se stejným *Jednozn. řetězcem*. Pokud je vše v pořádku, vyplatí *B* peníze a příkaz archivuje.
10. Pokud příkaz se stejným *Jednozn. řetězcem* již banka přijala provede zkoumání rozkrytých identifikačních řetězců. Jsou-li v obou případech stejný, podvádí obchodník, jinak podvádí *A*.

Protokol zajišťuje, že obchodník ani *A* nemohou podvádět. Musí však věřit bance, že s jejich účty nakládá korektně. Na druhou stranu pokud se oba chovají korektně, banka nemá k dispozici žádnou informaci o tom, jak nakládají se svými finančními prostředky.

Prahová schémata (treshold schemes)

protokol, k jehož úspěšnému provedení musí spojit síly více (např. *t*) účastníků

t nazýváme prahem (threshold value)
pokud *t* – 1 a méně účastníků nemůže získat ani částečný výsledek, mluvíme o *perfektním schématu*

Jednoduché autentizační schéma (2, 2)

nejprve „normální“ algoritmus:

odesílatel a příjemce sdílejí tajný klíč (*a, b*)
autentizační kód zprávy *m* získáme:

$$c = (am + b) \bmod p$$

příjemce přijme *m'* , spočítá

$$c' = (am' + b) \bmod p$$

a ověří, zda *c* = *c'*

totéž jako prahové schéma (2, 2)

a nahradíme dvěma stíny (shadow) s_1, s_2 tak, že $s_1 = a - s_2 \bmod p$, náhodně zvoleno tak, že $0 \le s_2 < p$, obdobně *b* nahradíme dvojicí s'_1, s'_2
zatímco příjemce si ponechá dvojici *a*, *b*, odesíletelé dostanou dvojici s_1, s'_1 resp. s_2, s'_2.
na vytvoření autentizačního kódu *c* nyní musí spolupracovat:

$$c_1 = (s_1 m + s'_1) \bmod p \text{ resp. } c_2 = (s_2 m + s'_2) \bmod p$$

příjemce sečte obě hodnoty (mod *p*) a dále je protokol stejný, jako předchozí.

uvedené schéma slouží pouze jako ilustrační případ, není bezpečné při vícenásobném použití

Sun a Shieh (t, n) prahové schéma

vedoucí zveřejní velké prvočíslo *p* a generátor α tělesa *T* mod *p*
každý účastník (shadowholder) H_i náhodně zvolí tajný stín s_i a spočítá svůj veřejný parametr

$$p_i = \alpha^{s_i} \bmod p$$

vedoucí následně zvolí (a uchová v tajnosti) náhodný polynom stupně $\mathbf{f}_0(X)$ stupně *t* – 1 nad *T* takový, že jeho konstantní člen je roven sdílené informaci I_0 a náhodný prvek k_0 tělesa *T*.

Na závěr publikuje parametr

$$R_0 = \alpha^{k_0} \bmod p$$

a hodnoty

$$C_{0i} = \mathbf{f}_0(i) \times (p_i)^{k_0} \bmod p$$

Tím je schéma hotovo.
Pokud dojde k *R*-té změně sdílené informace, vedoucí zvolí nový náhodný prvek k_R tělesa *T* a nový polynom $\mathbf{f}_R(X)$ a znovu vypočítá parametr R_R a hodnoty C_{Ri}.

nechť účastníci H_1 až H_t chtějí sestavit informaci I_R
každý z nich spočítá svůj opravdový stín (true shadow)

$$\mathbf{f}_R(i) = C_{Ri} \times (R_R)^{-s_i} \bmod p$$

a zveřejní jej.
tím získáme *t* bodů $(i, \mathbf{f}_R(i))$ a Lagrangeovou interpolací můžeme spočítat původní polynom $\mathbf{f}_R(X)$

Původní Sun a Shieh schéma navíc obsahuje mechanismus pro detekci útočníků, kteří by se pokoušeli podsouvat vadný opravdový stín. Tento mechanismus jsme pro jednoduchost vypustili, ostatně stejně za jistých okolností způsobuje snížení prahu schématu.

11 dobře míněných rad

1) Jasně definovat předpoklady o bezpečnosti
2) Osvětlit účel použité kryptografie: utajení, autentizace, binding,
náhodná čísla – nezaměňovat použití kryptografie s bezpečností či utajením
3) Postarat se o vzájemnou vazbu a pořadí jednotlivých zpráv
4) Je-li identita principála (pojmenovatelná entita = subjekt) podstatná pro interpretaci obsahu zprávy, musí být explicitně uvedena
5) Použijte dostatek redundance
6) Protokol by neměl využívat neověřené předpoklady o vlastnostech kryptograf. algoritmů, na nichž je založen
7) Je-li podpis počítán na základě zašifrovaných dat, nemůže sloužit k ověření, že druhá strana zná jejich obsah
8) Nevěřte v bezpečnost tajných informací druhých

9) Když podepisujete nebo dešifrujete data, pozor aby vás oponent nemohl používat jako orákulum (někdo kdo zná odpovědi)
10) Nemíchat dešifrování s podpisem
11) Zajistěte rozlišení různých instancí téhož protokolu

Princip explicitnosti
Robustnost bezpečnosti spočívá v explicitnosti
- jména, typy, časové známky
- v návrhu počátečních předpokladů, cílů, seznamu vlastností využitelných pro útok