MAC

Matej Zavrtanik

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
matej.zavrtanik@gmail.com

8. februar 2017

Povzetek

This paper briefly examines different algorithms to compute MAC (message authentication code) tags and possible attacks. MAC tags are used to verify the integrity of messages. In second chapter paper shortly describes cryptographic hash function and their relevance to compute MAC tags. In third chapter simple MAC scheme and three algorithms CBC-MAC, HMAC and Poly1305 are briefly described. Then three possible schemes to use MAC algorithms on plaintext or ciphertext are examined. In fifth chapter a benchmark compares speed of different cryptographic primitives.

I. Uvod

Z uporabo MAC (ang. message authentication code) dosežemo pristnost in integriteto poslanih sporočil. Z uporabo simetričnih šifer dosežemo zgolj zaupnost sporočil, ne pa tudi pristnosti in integritete. S pravilno uporabo MAC v navezi s simetričnim šifriranjem ali pa z uporabo bločnega načina overjenega šifriranja dosežemo tudi pristnost in integriteto.

Pomembnost simetričnih šifer je v zagotavljanju zaupnosti sporočil in s tem obrambo proti pasivnem napadalcu, ki ima možnost prestrezanja sporočil, ne pa tudi spreminjanja prestreženih oziroma pošiljanja poljubnih sporočil. Če si želimo zagotoviti varnost pred aktivnim napadalcem pa je uporaba overjenega šifriranja nujna, saj le tako zaznamo spremembe poslanih sporočil.

MAC značko lahko izračunamo na različne načine, najpogostejša sta z uporabo simetričnih bločnih šifer ali pa z uporabo zgoščevalnih funkcij. V obeh primerih za izračun MAC značke potrebujemo skupen ključ. Najbolj znan in razširjen način je z uporabo HMAC (ang. keyed-hash message authentication code), ki je tudi standariziran. Algoritmi za izračun MAC značke predpisujejo kako uporabiti simetrično šifro ali zgoščevalno funkcijo, ne pa tudi tega ali jih uporabimo na čistopisu ali na šifriranemu sporočilu. Zato ni pomemben zgolj izbor MAC algoritma, ampak tudi način uporabe.

MAC so uporabljajo v podobne namene kot digitalni podpisi. Glavne razlike so v tem, da za izračun MAC značke se uporablja skupen ključ, pri digitalnih podpisih pa asimetrična kriptografija. In pri MAC značkah se za ključ dogovorimo na začetku seje.

V II. poglavju so na kratko opisane zgoščevalne funkcije, saj se pogosto uporabljajo za izračun MAC značk. V III. poglavju so opisani algoritmi za izračun MAC značk in nekateri napadi. Nato so v IV. poglavju opisani trije načini uporabe algoritmov za izračun MAC značk. Primerjava hitrosti zgoščevalnih funkcij in simetričnih šifer je v V. poglavju. V zadnjem - VI. poglavju je podan zaključek.

II. Zgoščevalne funkcije

Za izračun MAC značke se pogosto uporabljajo kriptografske zgoščevalne funkcije. Zgoščevalne funkcije so primerne za izračun MAC značk iz več razlogov. Prvič, za zgoščevalne funkcije velja, da so hitre. Drugič, z njimi je relativno enostavno izračunati MAC značke.

i. CRC

Primer zgoščevalne funkcije, ki se uporablja za zaznavanje napak pri prenosu podatkov je CRC (ang. cyclic redundancy check). Zaradi premajhnega izhoda funkcije in zaradi linearnosti, je CRC neprimerna funkcija za kriptografske namene [2]. Funkcija CRC-16 se uporablja za detekcijo napak pri protokolu TCP.

Ker je funkcija CRC linearna nad operacijo ekskluzivnim ali-jem, velja $CRC(x \oplus y) = CRC(x) \oplus CRC(y)$. V primeru, da uporabimo CRC za integriteto sporočil kriptiranih s tekočo šifro kot je npr. RC4, za katero velja $E(k, x) \oplus y = E(k, x \oplus y)$. Potem velja tudi $E(k, CRC(x)) \oplus CRC(y) = E(k, CRC(x \oplus y))$. Zato za uspešen napad moramo poznati le mesto, kjer se nahaja izračunan CRC v šifriranemu sporočilu. Te pomanjkljivosti so izkoristili pri napadu na protokol WEP. [10].

ii. Kriptografske zgoščevalne funkcije

Med bolj znane in varne zgoščevalne funkcije za uporabo spadajo: SHA-2, SHA-3, SKEIN in BLAKE. Še nedolgo nazaj sta se veliko uporabljali tudi MD5 in SHA-1, ki pa nista več varni in zato primerni za uporabo v kriptografske namene. Na varnost zgoščevalne funkcije vpliva velikost izhoda. V kolikor je ta premajhen, kot v primeru MD5, za katero je izhod velikosti 128 bitov. Potem je mogoče poiskati dva enaka izhoda po približno $1.2 \cdot 2^l$ izračunanih vrednosti zgoščevalne funkcije z velikostjo izhoda l bitov. Za MD5 to pomeni, da bi po približno 2^{77} različnih vstavljenih vhodih dobili dva enaka enaka izhoda.

Zgoščevalna funkcija pa mora biti odporna tudi proti težje izvedljivemu "first pre-image attack". Da napad ni praktično izvedljiv mora biti težko poiskati tak x, da bo $H(x) = y$, za določen y. Ko izračunamo MAC značko s pomočjo zgoščevalne funkcije na način $H(k\|m) = t$, in kjer je k ključ, m sporočilo in t pripadajoča MAC značka. Je del vhoda v zgoščevalno funkcijo poznanega. Zato mora biti ključ dovolj velik, da ga ne uganemo na silo in zgoščevalna funkcija mora biti odporna na to, da ne izpeljemo ključa iz poznanih parov m in t.

III. MAC

Algoritem MAC (ang. message authentication code) lahko sestavimo s pomočjo simetričnih bločnih šifer ali pa z uporabo zgoščevalnih funkcij. Večina algoritmov za izračun MAC značk dopušča uporabo poljubnih kriptografskih funkcij. Npr. pri HMAC lahko uporabimo katerokoli zgoščevalno funkcijo, seveda pa bo varnost odvisna tudi od kvalitete zgoščevalne funkcije [6].

Algoritem za izračun MAC značk ni varen za uporabo, če obstaja način, da z enim ali več pridobljenimi pari sporočil in veljavnim MAC značk. Lahko sestavimo novo sporočilo, ki bo imelo veljavno MAC značko. Pri tem pa ni pomembno, ali ima vsebina novega sporočila smisel ali ne. Prav tako je MAC značka lahko enaka obstoječim ali pa je na novo izračunana. Posledično to pomeni tudi, da iz pridobljenih parov sporočil in MAC značk ne moremo pridobiti ključa.

i. Najbolj preprost izračun MAC značke

Najbolj preprost način za izračun MAC značke je z uporabo zgoščevalne funkcije. Za ključ k in sporočilo m jo izračunamo takole $t = H(k\|m)$, kjer je t veljavna MAC značka za sporočilo m.

Napad s podaljševanjem sporočila

Zgornja shema je ranljiva v primeru, da je zgoščevalna funkcija sestavljena s konstrukcijo Merkle-Damgård [5] [16]. V to skupino spadajo zgoščevalne funkcije MD5, SHA-1 in SHA-2. Kar so izkoristili pri napadu na spletno stran Flickr [11].

Napad je mogoče izvesti takole, pridobljena značka t, sporočila m se uporabi za določitev notranjega stanja zgoščevalne funkcije. Nato se na novo izračuna vrednost zgoščevalne funkcije t' za podaljšek m'. Značka t' bo veljavna za sporočilo $m\|m'$ s ključem k.

Novejše zgoščevalne funkcije kot je npr. SHA-3, so odporne na ta napad, zato bi MAC značko z uporabo SHA-3 lahko izračunali tudi na način $t = H(k\|m)$.

ii. MAC z ovojnico

MAC z ovojnico [14] uporablja zgoščevalno funkcijo za izračun MAC značke. Algoritem je definiran za zgoščevalno funkcijo MD5, lahko pa bi ga uporabili tudi za druge zgoščevalne funkcije. MAC značko t izračuna takole $t = H(k\|p\|m\|k)$, kjer zgoščevalna funkcija ključu pred sporočilom m, pripne bite p do konca velikosti bloka zgoščevalne funkcije. Temu je pripeto sporočilo. Sporočilo ni poravnano na velikost bloka, ampak dobi še enkrat pripet ključ. Na koncu so še enkrat pripeti biti, kot jih predpisuje zgoščevalna funkcija MD5.

Obstaja tudi MAC z ovojnico z dvema ključema, ki izračuna MAC značko na način: $t = H(k_1\|m\|k_2)$.

iii. CBC-MAC

CBC-MAC izračuna MAC značko z uporabo poljubne simetrične bločne šifre v načinu CBC [1]. Prednost te sheme je v tem, da je način delovanja CBC preprost in v kolikor ga ne najdemo v knjižnici, ga lahko implementiramo sami. Slabost sheme je ta, da mora biti dolžina sporočila za katerega želimo izračunati MAC, v naprej znana in določena [7]. Zato mora biti vsak ključ uporabljen zgolj za sporočila znane in določene dolžine. Značko CBC-MAC izračunamo tako, da sporočilo m šifriramo z ključem k, IV (ang. initialization vector) pa nastavimo na 0. Zadnji blok pri šifriranju je značka CBC-MAC.

Obstaja izboljšana verzija CBC-MAC imenovana ECBC-MAC, ki omogoča varno računanje MAC značke brez v naprej določene dolžine sporočila in je odporna na napad z kombiniranjem sporočil. ECBC-MAC je definiran takole: $ECBC - MAC(m, (k_1, k_2) = E(k_2, CBC - MAC(k_1, m))$.

Nepravilna uporaba in slaba implementacija CBC-MAC lahko pripelje do različnih napadov. Poleg dveh opisanih napadov obstaja še napad, ki je je mogoč v primeru, da tudi šifriramo v načinu CBC in pri tem uporabimo isti ključ za šifriranje in računanje MAC značke.

Napad z kombiniranjem sporočil

Če za CBC-MAC dovolimo, da so sporočila poljubne dolžine je mogoč napad z kombiniranjem sporočil [1] [7]. Za izvedbo napada je potrebno poznati dva para sporočil in veljavnih MAC značk. Za sporočili in pripadajoči znački (m, t) in (m', t'), lahko sestavimo novo sporočilo m'', za katero bo značka t' veljavna. Novo sporočilo sestavimo takole $m'' = m\|[(m'_1 \oplus t)\|m'_2\| \ldots \|m'_x]$.

Ena izmed možnih rešitev za preprečitev tega napada je uporaba polja z dolžino sporočila. Ampak tudi s tem se ne odpravi vseh težav, saj mora biti to polje na začetku sporočila, kar pomeni da moramo v naprej poznati dolžino sporočila.

Napad na poljuben IV

Napad je mogoč v primeru, da je IV različen od nič [1]. In da napadalec lahko prepriča eno izmed strani, da uporabi njegov IV. Pri tem napadu je mogoče poljubno spremeniti prvi blok. Če za sporočilo $M_1 = m_1|m_2 \ldots$ in IV_1 izračunamo CBC-MAC značko T_1. Potem lahko sestavimo sporočilo $M_2 = m'_1|m_2 \ldots$ in definiramo spremembo bitov $d = m_1 \oplus m'_1$. Zato ker je prvi blok sedaj $m'_1 = m_1 \oplus d$ in $IV'_1 = IV \oplus d$. Ko oboje združimo z operacijo ekskluzivnega ali nad biti se biti d-ja izničijo. Kar pomeni da je $m_1 \oplus IV_1 = m'_1 \oplus IV'_1$. Zato bo za obe sporočili značka T_1 veljavna.

iv. HMAC

Članka [6] in [8] ter RFC [12] opisujejo HMAC. HMAC je algoritem za MAC v kolikor ustreza zgornji definiciji. V kolikor MAC sestavimo z uporabo zgoščevalnih funkcij še ne pomeni, da je to HMAC. Saj obstaja več standariziranih načinov izračuna MAC značk s pomočjo zgoščevalnih funkcij. Prednost algoritma HMAC v primerjavi s CBC-MAC je v tem, da ni potrebno poznati dolžine sporočila v naprej.

NMAC

HMAC algoritem je izpeljan iz algoritma NMAC (ang. nested MAC). NMAC algoritem uporabi ključa za nastavitev IV zgoščevalne funkcije. V večini primerov se zgoščevalne funkcije uporablja brez spreminjanja IV, sicer za isto sporočilo dobimo drugačna izhoda z različnima IV. Velikost ključa je torej odvisna od velikosti IV oziroma notranjega stanja. To pa je vsaj toliko, kolikor je velik izhod zgoščevalne funkcije ali več. Za MD5 je to 128 bitov, za SHA-256 pa 256 bitov.

$NMAC_k(m) = H_{k_1}(H_{k_2}(m))$

Algoritem gnezdi zgoščevalni funkciji, da prepreči napad s podaljševanjem sporočila.

Algoritem HMAC

Slabost algoritma NMAC je v tem, da je potrebno spreminjati notranje stanje zgoščevalne funkcije. Ker notranjega stanja ni vedno mogoče spreminjati, je iz NMAC algoritma nastal algoritem HMAC. Algoritem HMAC ne spreminja notranjega stanja zgoščevalne funkcije, ampak vstavi ključ na vhod funkcije. HMAC uporablja zgolj en ključ za razliko od NMAC algoritma.

Za ključ k in sporočilo m je funkcija $HMAC_k(m)$ definirana takole:

$H(k \oplus opad \| H(k \oplus ipad \| m))$

Kjer je *opad* dolžine bloka in je enak $0X5C5C \ldots 5C$ in *ipad* je prav tako dolžine bloka in je enak $0X3636 \ldots 36$. Ker je ključ krajši od dolžine bloka zgoščevalne funkcije, so do konca bloka dodane ničle.

HMAC zelo pogosto uporablja SHA-256 za zgoščevalno funkcijo.

v. Poly1305

Algoritem je opisan v [15] in [9]. V opisu algoritma je uporabljena bločna šifra AES, kljub temu pa lahko uporabimo tudi druge bločne šifre. Število 1305 iz imena šifre je predstavlja $2^{130} - 5$ in je praštevilo, ki se uporablja za izračun MAC značke.

Šifra Poly1305 uporablja 16 bajtov velik ključ k, 16 bajtov velik dodaten ključ r in 16 bajtov velik števec (ang. nonce) n [9]. Števec je lahko javen, a se ne sme ponavljati in mora biti naključen. Pri tem pa velja omejitev pri izbiri r. Če z r_i označimo i-ti bajt r. Potem morajo biti zgornji štirje biti v bajtih r_3, r_7, r_{11} in r_{15} enaki nič in za bajte r_4, r_8 in r_{12} morajo biti spodnja dva bita enaka nič. Kar zmanjša r na 106 bitov.

Izračun MAC značke za sporočilo m dolžine l bajtov, kjer je m_i i-ti bajt sporočila, poteka takole. Izračuna se $q = \lceil l/16 \rceil$. Nato izračunamo števila $c_1, c_2, \ldots, c_q \in \{1, 2, 3, \ldots, 2^{129}\}$. Potem za $1 \leq i \leq \lfloor l/16 \rfloor$, izračunamo:

$c_i = m_{16i-16} + 2^8 m_{16i-15} + \cdots + 2^{120} m_{16i-1} + 2^{128}$

Če število l ni deljivo s 16 je izračun nekoliko drugačen. V vsakem primeru je 16 bajtov velik blok sporočila razširjen na 17 bajtov.

Sporočilo se overi z uporabo naslednje formule:

$(((c_1 r^q + c_2 r^{q-1} + \cdots + c_q r^1) \mod 2^{130} - 5) + AES_k(n) \mod 2^{128}$

Danes je Poly1305 široko uporabljan, saj ga je podjetje Google izbralo skupaj s tekočo šifro ChaCha20 za protokol TLS [13].

vi. Overjeno šifriranje

Obstaja tudi načini šifriranja s simetrično bločno šifro, ki imajo vgrajeno overjanje. V to kategorijo spadajo GCM (ang. Galois/counter mode), OCB (ang. Offset codebook mode) in EAX. Izmed naštetih se največ uporablja način GCM skupaj z algoritmom AES. Zato se dodatnemu računanju MAC značk lahko izognemo, če uporabimo enega izmed naštetih načinov. Žal pa teh načinov delovanja ne moremo uporabiti pri tekočih šifrah.

IV. Načini uporabe MAC značk

MAC značke lahko izračunamo na več različnih načinov [8]. Lahko jih izračunamo tudi na čistopisu, ki bo poslan nešifriran in s tem zagotovimo integriteto sporočila. V kolikor pa jih izračunamo za sporočila, ki bodo šifrirana, se pojavi vprašanje ali naj bo MAC značka izračunana nad čistopisom ali nad šifriranim sporočilom. Oziroma ali naj bo MAC značka šifrirana

ali ne. Z oznako ε označujemo simetrično šifro in z k_e uporabljen ključ. Z oznako τ pa označujemo funkcijo za izračun MAC značk in z k_m njen ključ. Zaradi nekaterih napadov je bolje, da sta ključa različna.

V članku možnost izračuna MAC značke tako za čistopis, kot za šifrirano sporočilo ni bila omenjena. V vsakem primeru bi bila taka shema potratna, saj bi dvakrat računali MAC značko in tudi pri varnosti ne bi pridobili. Glede na to, da shema šifriraj in potem MAC zagotavlja integriteto čistopisa in šifriranega sporočila.

i. Šifriraj in MAC

Pri tej shemi izračunamo MAC značko nad čistopisom, priložena pa je šifriranemu sporočilu.

$\bar{\varepsilon}(k_e \| k_m, m) = \varepsilon(k_e, m) \| \tau(k_m, m)$

Slabost te sheme je v tem, da zagotavlja integriteto čistopisa, ne pa tudi šifriranega sporočila. Da odkrijemo napako, je potrebno šifrirano sporočilo dešifrirati in šele nato lahko preverimo integriteto. Zato v primeru, da se za šifriranje uporabi bločno šifro v načinu CBC Je ta shema ranljiva na "Padding oracle attack"[3].

Ta shema pa ima veliko pomanjkljivost. Napadalec, ki prestreza šifrirana sporočila in MAC značke, lahko ugotovi, da sta bili poslani dve sporočili z enako vsebino na podlagi enakih MAC značk. Zato bi bilo pri uporabi te sheme smiselno uporabiti števec, ki bi bil dodan sporočilu pri računanju MAC značke. Števec pa lahko ostane javen in je del MAC značke. Namen števca je izključno onemogočiti napadalcu, da bi prepoznal dve različni šifrirani sporočili z enakim čistopisom.

Ta shema z manjšo spremembo, se uporablja v protokolu SSH.

ii. MAC in potem šifriraj

V tem primeru se MAC značka izračuna nad čistopisom in nato pa je šifrirana skupaj s čistopisom. Tudi ta shema nudi zgolj integriteto čistopisa, ne pa tudi integritete šifriranega sporočila. Tudi ta shema je ranljiva na "Padding oracle attack", če se za šifriranje uporabi bločno šifro v načinu CBC [3].

$\bar{\varepsilon}(k_e \| k_m, m) = \varepsilon(k_e, m \| \tau(k_m, m))$

Prednost te sheme je v tem, da napadalec nikoli ne pridobi parov sporočil z veljavnimi MAC značkami. Za razliko od preostalih dveh shem, kjer bi ključ lahko izračunali na silo, če bi le-ta bil dovolj majhen.

Ta shema se je uporabljala v protokolu SSL.

iii. Šifriraj in potem MAC

Za razliko od prejšnjih dveh shem ta najprej šifrira, nato pa izračuna MAC značko nad šifriranim sporočilom. Ta shema zagotavlja integriteto šifriranega sporočila in s tem tudi čistopisa.

$\bar{\varepsilon}(k_e \| k_m, m) = C \| \tau(k_m, C)$, kjer je $C = \varepsilon(k_e, m)$

Ta način izračuna MAC značke velja za najboljšega s stališča varnosti [8]. Z uporabo tega načina lahko preverimo integriteto šifriranega sporočila in s tem dosežemo tudi integriteto čistopisa. Zato prejemnik sporočila zazna spremembe preden se loti dešifriranja, kar pomeni, da je shema varna proti "Padding oracle attack". V vsakem primeru pa si prihranimo dešifriranje sporočila, v kolikor preverjanje integritete ne uspe.

Ta shema se uporablja v protokolu IPSec.

V. Primerjava hitrosti

Ker se danes večina prometa na spletu šifrira, je pomembna tudi hitrost šifriranja. Novejši procesorji imajo podporo ukazom AES-NI, ki bistveno pohitri delovanje šifre AES. Test je bil opravljen v programu LibreSSL različice 2.5.1 na računalniku s procesorjem Intel(R) Core(TM) i5-4570 in operacijskim sistemom Ubuntu 16.04. Hitrost kriptografskih funkcij je bila pomembna na natečaju za AES in SHA-3. V tabeli 1 je primerjava hitrosti med zgoščevalno funkcijo SHA-2, bločnima šiframa AES in Camellia, ter tekočo šifro ChaCha20.

Iz rezultatov je očitno, da je AES v načinu delovanja GCM daleč najhitrejša. Razlog za hitrost šifre AES v načinu GCM je v tem, da

Šifra / vhod	1024 b	8192 b
SHA-256	263 MB/s	276 MB/s
SHA-512	394 MB/s	432 MB/s
AES-128-GCM	1532 MB/s	1724 MB/s
AES-128-CBC	159 MB/s	160 MB/s
Camellia-128-CBC	192 MB/s	195 MB/s
ChaCha20-Poly1305	327 MB/s	355 MB/s

Tabela 1: *Hitrost računanja kriptografskih algoritmov v eni sekundi glede na velikost vhoda.*

uporablja ukaze AES-NI in da GCM način omogoča paralelizacijo šifriranja. Medtem ko AES v načinu CBC ne uporablja ukazov AES-NI (razlog za to ni znan) in zaradi samega delovanja CBC ni mogoče paralelizirati. Programska implementacija AES v načinu CBC je celo počasnejša od šifre Camellia. Zgoščevalna funkcija SHA-2 je sicer hitrejša od programske izvedbe AES, a ne nujno tudi od tekoče šifre ChaCha20 skupaj z MAC algoritmom Poly1305.

Pri tem pa velja omeniti, da AES-128-GCM izkorišča štiri jedra, medtem ko preostale šifre zgolj eno. Zato bi bila prepustnost preostalih šifer približno štirikrat večja, če bi vsa jedra šifrirala različna sporočila.

Žal rezultatov za AES v načinu CTR ni bilo mogoče pridobiti, saj knjižnica ne podpira tega načina delovanja. Načina CTR sicer ni mogoče uporabiti za izračun MAC značke, a velja za enega izmed najhitrejših načinov delovanja bločnih šifer.

Na podlagi tega testa lahko sklepamo, da je AES/GCM boljša izbira kar se hitrosti tiče od preostalih algoritmov za doseganje overjenega šifriranja. Pri preostalih šifrah moramo upoštevati tudi to, da v primeru šifriranja sporočila z AES v načinu CBC, moramo temu dodati še računanje MAC značke.

VI. Zaključek

V članku so opisani algoritmi za računanje MAC značk in njihova uporaba skupaj s šifriranjem. Številni napadi potrjujejo, da overjanje in zagotavljanje integritete sporočil ni tako preprosto kot se na prvi pogled zdi. Pomembno je, da uporabimo standarizirane rešitve kot sta HMAC ali Poly1305. Sicer lahko hitro zaidemo v težave, kot so programerji pri podjetju Flickr, ki so omogočili napad s podaljševanjem.

V zadnjih nekaj letih ni bilo novih algoritmov, ki bi izpodrinili te, ki so v uporabi za izračun MAC značk. Še najnovejši je Poly1305, ki tudi že star več kot 10 let. Prihodnost je najbrž v načinih delovanja, ki imajo vgrajeno overjanje kot je npr. GCM. Tudi iz testov V. poglavja, kjer so primerjani različne šifre po prepustnosti, se način delovanja GCM izkaže za najhitrejšega, poleg tega pa nudi enak nivo varnosti kor preostali načini šifriranja s pravilnim načinom računanja MAC značk.

Kljub temu se bodo MAC značke uporabljale še nekaj časa, saj nekateri protokoli za sporočila ne zahtevajo zaupnosti le-teh.

Literatura

[1] CBC-MAC. Wikipedia.

[2] Cyclic redundancy check. Wikipedia.

[3] Padding oracle attack. Wikipedia.

[4] Bellare, M. New proofs for nmac and hmac: Security without collision resistance. *J. Cryptol. 28*, 4 (Oct. 2015), 844–878.

[5] Bellare, M., Canetti, R., and Krawczyk, H. Keying hash functions for message authentication. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology* (London, UK, UK, 1996), CRYPTO '96, Springer-Verlag, pp. 1–15.

[6] Bellare, M., Canetti, R., and Krawczyk, H. Message Authentication using Hash Functions- The HMAC Construction. *CryptoBytes 2* (1996).

[7] Bellare, M., Kilian, J., and Rogaway, P. The security of the cipher block chaining message authentication code. *J. Comput. Syst. Sci. 61*, 3 (Dec. 2000), 362–399.

[8] Bellare, M., and Namprempre, C. Authenticated encryption: Relations

among notions and analysis of the generic composition paradigm. Springer-Verlag, pp. 531–545.

[9] Bernstein, D. J. The poly1305-aes message-authentication code. In *Proceedings of the 12th International Conference on Fast Software Encryption* (Berlin, Heidelberg, 2005), FSE'05, Springer-Verlag, pp. 32–49.

[10] Cam-winget, N., Housley, R., Wagner, D., and Walker, J. Security flaws in 802.11 data link protocols, 2003.

[11] Duong, T., and Rizzo, J. Flickr's api signature forgery vulnerability, 2009.

[12] Krawczyk, H., Bellare, M., and Canetti, R. HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Informational), Feb. 1997. Updated by RFC 6151.

[13] Langley, A., Chang, W., Mavrogiannopoulos, N., Strombergson, J., and Josefsson, S. ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS). RFC 7905 (Proposed Standard), June 2016.

[14] Metzger, P., and Simpson, W. IP Authentication using Keyed MD5. RFC 1828 (Historic), Aug. 1995.

[15] Nir, Y., and Langley, A. ChaCha20 and Poly1305 for IETF Protocols. RFC 7539 (Informational), May 2015.

[16] Tsudik, G. Message authentication with one-way hash functions. *SIGCOMM Comput. Commun. Rev. 22*, 5 (Oct. 1992), 29–38.