Twisted Polynomials and Forgery Attacks on GCM

Mohamed Ahmed Abdelraheem, Peter Beelen,
Andrey Bogdanov, Elmar Tischhauser

Department of Applied Mathematics and Computer Science
Technical University of Denmark

Eurocrypt 2015

Outline

Polynomial Hashing Authentication

Polynomial Hashing Scheme

Processes an input consisting of a key H and plaintext/ciphertext $M = (M_1||M_2||\cdots||M_t)$, where each $M_i \in \mathbb{F}_2^{128}$, by evaluating

$$h_H(M) := \sum_{i=1}^{t} M_i H^i \in \mathbb{F}_2^{128}$$

Poly hashing is used:

- GCM (NIST standard).
- Some CAESAR candidates (Ongoing Crypto Competition).

Outline

Recent Attack: Procter' and Cid's General Framework (FSE 2013)

Previous attacks are special cases of this framework.

Forgery Polynomial

Assume that $q(X) = \sum_{i=1}^{r} q_i X^i$ and that $q(H) = 0$. Assume that $M = (M_1||M_2||\cdots||M_l), Q = (q_1||q_2||\cdots||q_r)$ and that $l < r$. Then

$$\begin{aligned} h_H(M) &= \sum_{i=1}^{r} M_i H^i = \sum_{i=1}^{l} M_i H^i + \sum_{i=1}^{r} q_i H^i \\ &= \sum_{i=1}^{r} (M_i + q_i) H^i \\ &= h_H(M + Q) \end{aligned}$$

If $h_H(M) = h_H(M + Q)$ then H might be a **weak key**.

Handschuh and Preneel's Weak-key Definition (Crypto 2008)

Weak Keys

A class of keys of size N is called **weak** if membership testing:

- costs **less than** N key tests and verification queries.

Outline

Open Questions

Question I

How to efficiently construct a forgery polynomial $q(X)$ of degree t?

- **Naïve way** $(X - H_1) \cdots (X - H_t)$: impractical.
- **random polynomials:** repeated roots.

Only proposed explicit polynomials:

- $X^{t+1} - X$ where $t | (2^{128} - 1)$ due to Saarinen (FSE 2012).
- Not useful for efficient key recovery.

Question II

How to efficiently cover the whole key space with a set of forgery polynomials?

Goal: explicit non-cyclic polynomials with unique roots from different subgroups covering key space.

Outline

Twisted Polynomials

- Let V be a vector subspace of $\mathbb{F}_{2^{128}}$ with d-dim basis
- Polynomials $p_V(X)$ whose **roots are exactly the elements of** V can be constructed efficiently using **Ore ring theory**.
- **Twisted polynomials** of degree 2^d but sparse with $d+1$ nonzeros

$$p_V(X) = X^{2^d} + c_{d-1}X^{2^{d-1}} + \cdots + c_1X^2 + c_0X$$

Remark

Ferguson (NIST Comment 2007) used **linearized polynomials** to recover GCM's hash key when short tags are used.

Outline

Disjoint Coverage

- $p_V(X - a) = p_V(X) - p_V(a)$
- $p_V(x) - p_V(a)$ has as sets of roots exactly $a + V$, $a \in \mathbb{F}_q$.
- Construct $n = 2^{128-d}$ polynomials with:
  - $a_1 + V, \cdots, a_n + V$ as roots to **disjointly cover** $\mathbb{F}_2^{128}$.

Example I

Forgery polynomial of degree 2^{31} but sparse with 31 nonzero coefficients ($c_i = a^{e_i}$):

i	e_i
1	576613647098987897394216259339443067 7
2	886405851238878607712823602816508493 69
3	228467699759147933517306066079059941262
4	608709206422113118601250588783762399 67
5	699813938596682643737860908514039195 97
6	255459844209463555435845538974500206397
7	263576500668765237830541241929740306586
8	371670151494514720087160030776564926 21
9	580432773787481077233241351194154844 05
10	321767455835401530567257366419614234023
11	450338884514507376214297123948464446 57
12	258425985086309803122357832308421510564
13	105831989526232747717837668269825340779
14	267464360177071876386745024557199320756
15	280644372754658909872880662034708629284
16	105000326856250697615431403289357708609
17	458258183594606115422832253689081928 57
18	828459613081692598766012671274594169 89
19	442179899361942084725223538212208611 15
20	690629439605523090898429831294031742 17
21	268462019404836089359334939776220681511
22	300016489421132402121135552937497655 14
23	669737854382487997736546203881056449
24	127958856468256956044189872000451203235
25	277162238678239965835219683143318848400
26	134662498954166373112542807113066342554
27	219278415175240762588240883266619436470
28	216197476010311230105259534730909158682
29	281783005767613667130380044536264251829
30	181483131639777656403198412151415404929
31	383848366876114263330516022408845847 92
32	0

Example II

Forgery polynomial of degree 2^{61} but sparse with 61 nonzero coefficients ($c_i = a^{e_i}$):

i	e_i	i	e_i
1	2052696313502677311977152941999124732 7	32	109604555581389038896555752982244394616
2	2645468516910265402517226187192457775 04	33	119482829110451460647031381779266776526
3	7927973230583347490289364796772159492 1	34	165259785861038013124994816644344468967
4	3257125555859085422915375601818696323 51	35	155444340258770748055544634836807134293
5	2811408387984342035893248854756124991 3	36	869821844387300458212740258310619614 30
6	2711479434514425475726752832034933257 75	37	104870645496065737272877350967826010844
7	3352555208237332520203924884077314323 38	38	562812815790023183370379193561271053 69
8	6718016882907633170860567569329895273	39	100068518982837928471870587740499831 41
9	2558890659818838679030196219910131254 35	40	936879200755548123588902448980883454 49
10	4945768772160146371264018921775547423 0	41	698326729003034322484017536582625335 06
11	3115790054425697302770307552286836168 07	42	246360754285298743574294101515912517720
12	2279845104054619648939249132688090663 93	43	895678936019042717674614594480764049 68
13	3246609530451183282355389001619979921 61	44	337681726780870315172220356080972321854
14	1013700597457892851275193977904942154 41	45	210317547004302372764274348440690947691
15	3358407778371420475556500752443734197 08	46	158574321133010145534802861165087620178
16	3145884998026720146174734707171090752 3	47	291559826228649927512447763293001897434
17	3394778189769142429629606542865477020 07	48	156351243312442316097609527177914577 46
18	2670562444913309576186854437219791202 06	49	196562458398036090488379086660199368109
19	1152743276516193470460917939924320071 52	50	308779188958300135859037769338975723488
20	3096064718383326108684543694831059048 88	51	311961723579011854596575128443762996895
21	3147283196347054338049354349673292976 3	52	153505386496968503239745640447605550270
22	1913325955971934246263223290320563780 09	53	266880473479137548264080346617303001989
23	1895539134313092556145141635506700756 72	54	325361660912502344542873376867973189476
24	2246173220526712483192578270674747408 67	55	756486261013747940931759163320432850 57
25	6304123030678803297311114553330705156 2	56	122904035765598179315104311504496672627
26	2215766062721523541533507393750403372 39	57	240654849065616783877381099532333510366
27	2917999035400062892202450451885737411 92	58	717747464603164639815429745582806718 65
28	2904896244379507644997072326197701862 93	59	318833970371431372762935716012099244730
29	2637547265060466399854792406606037770 00	60	176351990917361872511208705771673004140
30	4516080743616730799068915079205267070 7	61	227372417807158122619428517134408021585
31	3363088190599663092523770162295042595 0	62	0

Outline

Description of GCM

Ciphertext generation:

$$C_i = E_k(J_i) \oplus M_i$$

Tag generation:

$$T = E_k(J_0) \oplus h_H(C)$$

Counter values depends on the nonce N:

$$J_0 = \begin{cases} N||0^{31}||1 & \text{if } |\mathbf{N}| = \mathbf{96}, \\ h_H(N||0^{s+64}||[|N|]_{64}) & \text{if } |\mathbf{N}| \neq \mathbf{96}, \end{cases}$$

where $J_i = \mathrm{inc}_{32}(J_{i-1})$ and $s = 128\lceil |N|/128 \rceil - |N|$

Weak-key Detection Process on GCM

The weak-key detection process is as follows:

- **Observe** a ciphertext/tag pair $(C; T)$.
- **Submit** a verification query with 2^{31}-block $(C + Q; T)$.
- **Success:** H is root of $p_V(X)$ and we recover it using binary search key recovery (31 queries).

Universal Forgeries

Recovering H leads to nonce-respecting universal forgeries on GCM

Attack I: Slide Attack on GCM when $|N| \neq 96$ I (Chosen Nonce)

H **is known**. **Observe** M/C with nonce N.

Create a universal forgery for M' by **sliding** J_{i+1} into J'_i as follows:

- **Compute** $J_0 = h_H(N||0^{s+64}||[|N|]_{64})$, $J_{i+1} = \mathrm{inc}_{32}(J_i)$.
- **Compute** $E_K(J_i) = M_i \oplus C_i$.
- **Solve for** N'**:** $J'_0 = J_1 = h_H(N'||0^{s+64}||[|N'|]_{64})$
- **Compute** $C'_i = M'_i \oplus E_K(J'_i) = M'_i \oplus E_K(J_{i+1})$
- **Compute** $T' = E_K(J'_0) \oplus h_H(C') = E_K(J_1) \oplus h_H(C')$.
- **Output:** C' and T'.

Attack II: Interaction Attack on GCM when $|N| = 96$

Interaction possibility is one of the **undesirable characteristics** of GCM (Rogaway CRYPTREC Report 2011).

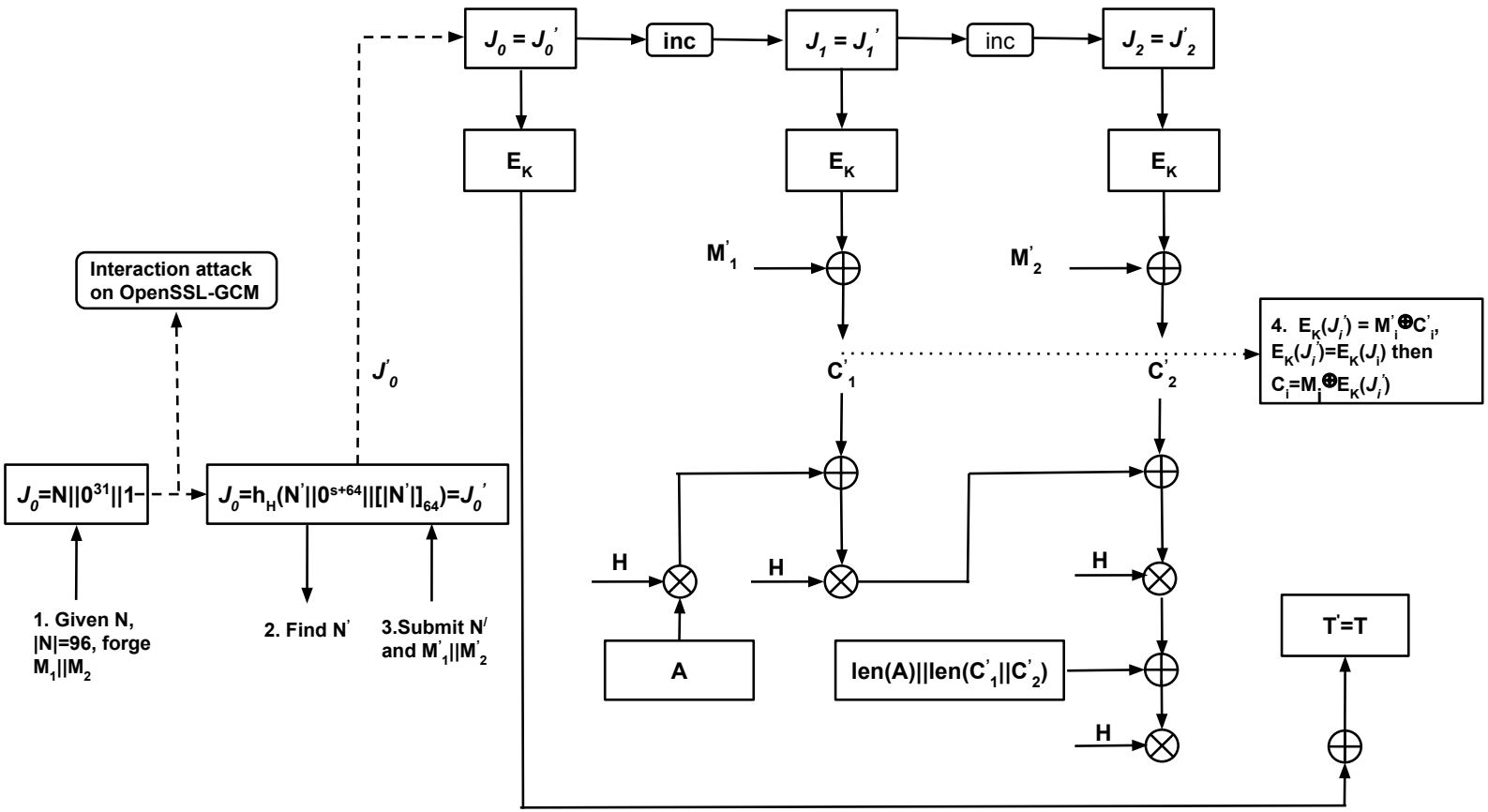

Figure: Forgeries for GCM via cross-nonce interaction

Further Results

- **Universal forgeries** on POET-(Galois mult. variant) (**withdrawn from CAESAR upon our analysis**)
- **Improved key recovery:**
  - Handschuh and Preneel's used verification queries (binary search) to recover GCM's hash key from an identified class of weak keys (Crypto 2008).
- **Universal forgeries** on COBRA and Julius.

Summary

- **Complete disjoint coverage** of the key space by forgery polynomials.
- **Nonce-respecting universal forgeries** weak-key attacks on **GCM**.
- **Universal fogeries** weak-key attacks on POET, COBRA and Julius.

Thank you for your attention