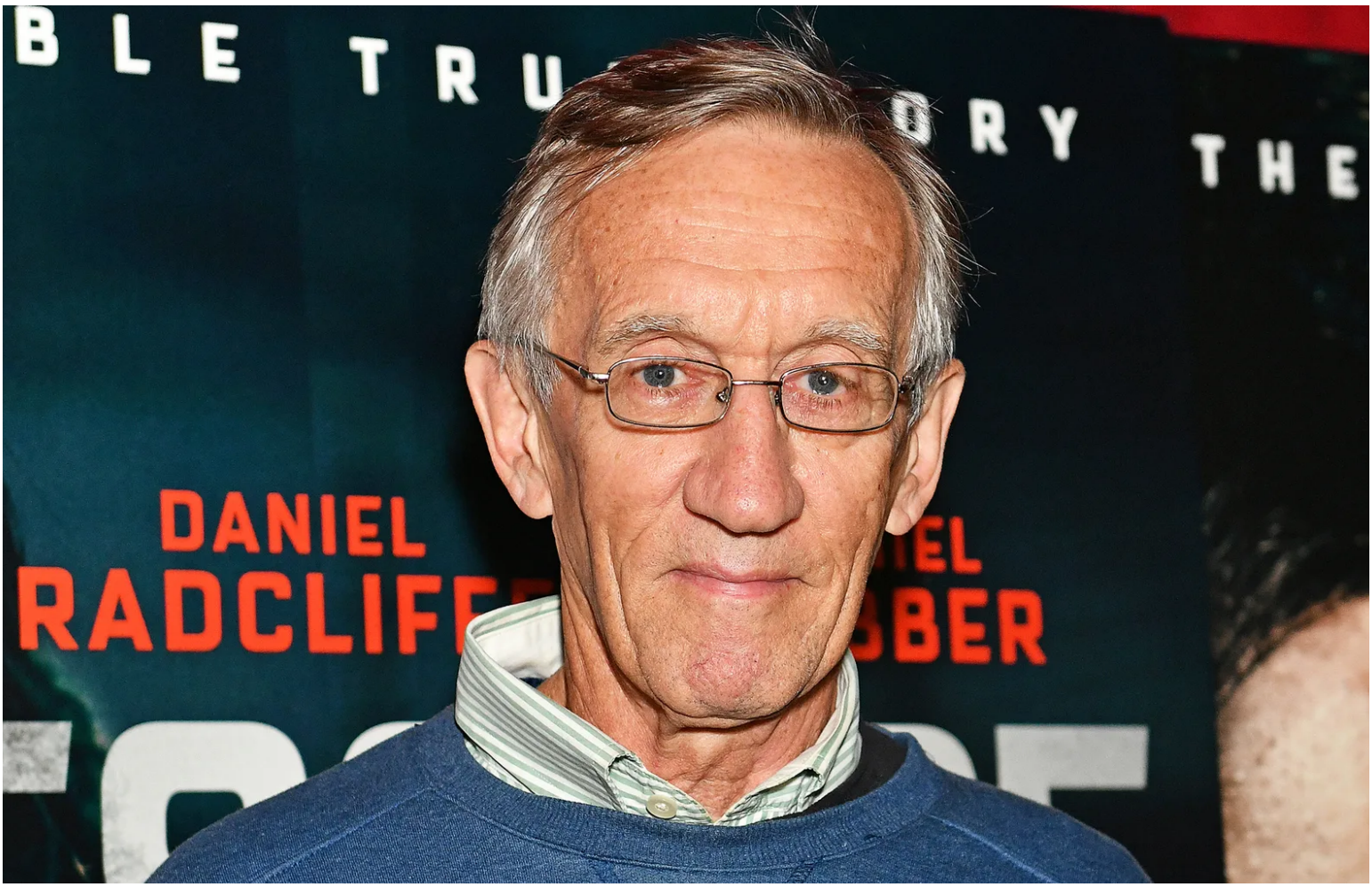
STEVEN LEVY BUSINESS OCT 18, 2024 9:00 AM

You Can Now See the Code That Helped End Apartheid

John Graham-Cumming, who happens to be Cloudflare's CTO, cracked a 30-year-old encrypted file that had a role in rewriting South Africa's history.



Tim Jenkin came up with a system that helped members of the African National Congress communicate safely under apartheid. PHOTOGRAPH: DAVE BENETT/GETTY IMAGES

n Graham-Cumming doesn't ping me often, but when he does I pay ention. His day job is the CTO of the security giant Cloudflare, but he is also a lay historian of technology, guided by a righteous compass. He might be best known for successfully leading a campaign to force the UK government to apologize to the legendary computer scientist Alan Turing for prosecuting him for homosexuality and essentially harassing him to death. So when he DM'd me to say that he had "a hell of a story" —promising "one-time pads! 8-bit computers! Flight attendants smuggling floppies full of random numbers into South Africa!"—I responded.

The story he shared centers around Tim Jenkin, a former anti-apartheid activist. Jenkin grew up "as a regular racist white South African," as he described it when I contacted him. But when Jenkin traveled abroad—beyond the filters of the police-state government—he learned about the brutal oppression in his home country, and in 1974 he offered his help to the African National Congress, the banned organization trying to overthrow the white regime. He returned to South Africa and engaged as an activist, distributing pamphlets. He had always had a penchant for gadgetry and was skilled in creating "leaflet bombs"—devices placed on the street that, when triggered, shot anti-government flyers into the air

to be spread by the wind. Unfortunately, he says, in 1978 “we got nicked.” Jenkin was sentenced to 12 years in prison.

Jenkin has a hacker mind—even as a kid he was fiddling with gadgets, and as a teen he took apart and reassembled his motorcycle. Those skills proved his salvation. Working in the woodshop, he crafted mockups of the large keys that could unlock the prison doors. After months of surreptitious carpentry and testing, he and two colleagues walked out of the prison and eventually got to London.

It was the early 1980s, and the ANC’s efforts were flagging. The problem

was communications. Activists, especially ANC leaders, were under constant surveillance by South African officials. “The decision was taken to get leadership figures back into the country to be closer to the activists, but to do that they still had to be in touch with the outside,” says Jenkin, who was given a mandate to solve the problem. Rudimentary methods—like invisible ink and sending codes by touch-tone dials—weren’t terribly effective. They wanted a communication system that was computerized and unbreakable. The plan was dubbed Operation Vula.

Working in his small council flat in the Islington neighborhood in London—nicknamed GCHQ, after the top-secret British intelligence agency—Jenkins set about learning to code. It was the early days of PCs, and the equipment by today’s standards was laughably weak. Breakthroughs in public key cryptography had come out a few years earlier, but there was no easily available implementation. And Jenkin was suspicious of prepackaged cryptosystems, fearing they might harbor back doors that would provide governments access.

Using a Toshiba T1000 PC running an early version of MS-DOS, Jenkin wrote a system using the most secure form of crypto, a one-time pad, which scrambles messages character by character using a shared key that’s as long as the message itself. Using the program, an activist could type a message on a computer and encrypt it with a floppy disk containing the one-time pad of random numbers. The activist could then convert the encrypted text into audio signals and play them to a tape recorder, which would store them. Then, using a public phone, the activist could call, say, ANC leaders in London or Lusaka, Zambia, and play the tape. The recipient would use a modem with an acoustic coupler to capture the sounds, translate them back into digital signals, and decrypt the message with Jenkin’s program.

One potential problem was getting the materials—the disks and

computers—to Africa. The solution, as Graham-Cumming noted, was accomplished by enlisting a sympathetic Dutch flight attendant who routinely flew to Pretoria. “She didn't know what she was taking in, because everything was packaged up; we didn't talk about it at all,” says Jenkin. “She just volunteered to take the stuff, and she took in the laptops and acoustic modems and those sorts of things.”

Operation Vula gave the ANC the confidence to sneak some leaders back into the country to supervise anti-government actions, coordinating efforts with the top leaders abroad. The Vula coding system even made it possible for the ANC brain trust to establish contact with the incarcerated Nelson Mandela. He received local visitors who came in carrying books that hid the decrypted dispatches—another product of Jenkin’s MacGyver-esque powers. “We smuggled these specially doctored books—innocuous looking books, maybe about flowers or travel—with a secret hidden compartment in the cover,” says Jenkin. “If you knew how to do it, you could extract the message and put another one back in there.”

Jenkin’s system allowed countless messages to be sent securely, as the ANC reached closer to its goal of defeating apartheid. He is unaware of any instance where the authorities decoded a single communication. When the ANC was ultimately unbanned in 1991, it credited Operation Vula as a key factor in its victory. In April 1994, Nelson Mandela became the president of South Africa.

You might be thinking that Jenkin’s story is so amazing that someone should make a movie out of it. Someone already has—focusing on the prison break. It’s called *Escape From Pretoria* and stars Daniel Radcliffe as Jenkin. There’s also a short documentary about Jenkin and Operation

Vula. But until this year one thing had not been documented—Jenkin's artisanal cryptosystem.

That's where Graham-Cumming enters the picture. Years ago, he'd heard about Operation Vula and found the story fascinating. Earlier this year, he came across a mention of it and wondered—what happened to the code? He felt it should be open-sourced and uploaded to GitHub for all to see and play with. So he contacted Jenkin—and heard a sad story.

When Jenkin returned to South Africa in 1992, he had been worried about taking his tools with him, as some elements of the operation were still ongoing. "I didn't want to just walk in with all this communication equipment and have this coding wind up in their hands, so I compressed everything into single files, zipped it with passwords, and brought in the disks like that." He had no problem at the border. Eventually, people felt safe meeting face-to-face and no longer needed Jenkin's system. "Then life caught up with me," he says. "I got married, had kids and all that. And one day, I thought, 'Let me have a look at this thing again.' And I couldn't remember the password." Over the years, Jenkin and others tried to break the encryption, and failed.

Rather than being disappointed, Graham-Cumming was thrilled. "I've got to have a go at this," he told himself, and asked for the files.

When Graham-Cumming received them on May 20, he was encouraged that they were compressed and encrypted in the old encrypted PKZIP format. It had a known flaw you could exploit if you knew some part of the original unencrypted message. But you'd have to know where in the zipped file that text is represented. He asked if Jenkin had any unencrypted versions of the code files, and indeed there were a few. But they turned out to be different from what was in the zip file, so they weren't immediately helpful.

Graham-Cumming took a few days to think out his next attack. He

realized the zip file contained another zip file, and that since all he needed was the right original text for a specific part of the scrambled text, his best chance was using the first file name mentioned in the zip within the zip. “You could predict the very first bit of that zip file using that name,” he says. “And I knew the names he was using. I was like, ‘Oh, I'm gonna try out a name,’ and I wrote a little program to try it.” (This is a much simplified explanation—Graham-Cumming provides more details in a blog post.)

On May 29, Graham-Cumming ran the program and stepped away to eat a breakfast of scrambled eggs. Twenty-three minutes later, the program finished. He’d broken the encryption and unzipped the file. The workings of Jenkin’s cryptosystem were exposed. It had been nine days since he first exchanged emails with Jenkin.

The next step was to actually run the code, which Graham-Cumming did using an emulator of the ancient version of MS-DOS used in the Toshiba T1000. It worked perfectly. Jenkin had feared that a professional coder like Graham-Cumming might find his work hopelessly amateurish, but his reaction was quite the opposite. “I’m pretty amazed, given the limitations he had in terms of knowledge, in terms of hardware, that they built something that was pretty credible, especially for the time,” says Graham-Cumming. Even more impressive: It did a job in the wild.

Jenkin, who has spent the past few decades in South Africa as a computer programmer and web designer, has now uploaded the code to GitHub and open-sourced it. He plans to unzip and upload some of the messages exchanged in the ’80s that helped bring down apartheid.

“The code itself is a historical document,” says Graham-Cumming. “It wasn't like, ‘Oh, I'm going to create some theoretical crypto system.’ It was like, ‘I’ve got real activists, real people in danger. I need real communications, and I need to be practical.’” It’s also, as he promised

me, a hell of a story.

Time Travel

In November 2014, I wrote for Backchannel about Graham-Cumming's campaign to evoke an apology from the UK for its shameful actions against Alan Turing.

On September 10, Graham-Cumming was sick with the flu. He stayed in bed most of the day. Late in the afternoon, he dragged himself to his computer to check his email. Sitting there, in rumpled gym garb, he found the following message from one Kirsty McNeill, a person he did not know. The email signature, as well as the email domain, indicated an association with 10 Downing Street.

Graham-Cumming, even in his flu-addled state, knew that this might just be some prank. It wasn't hard to spoof an address, even from the Prime Minister's office. He Googled the telephone number in the signature. It was the switchboard to 10 Downing Street. He dialed, asked for Ms. McNeill, and was quickly connected. "We are doing the apology tonight," she told him. Was it all right if she read him the text? Somewhat stunned, he listened and approved.

Ten minutes later, his iPhone rang. "Hello, John, this is [Prime Minister] Gordon Brown," came a familiar voice. "I think you know why I'm calling you." Over the next few minutes the two chatted. Prime Minister Brown was not a politician of the oozing Tony Blair/Bill Clinton "feel your pain" school. Graham-Cumming admits to some of the same social awkwardness. So the two of them stumbled through a conversation in which Brown confessed that until the petition he had

not realized the government's role in persecuting and prosecuting one of its greatest war heroes. Within a half an hour, 10 Downing released the apology.

Ask Me One Thing

Jean-Daniel asks, "Can we train AI to spot and flag AI-generated content automatically? If so can we incorporate that as a default in search engines, phones, and PCs?"

Thanks for the question, Jean-Daniel. You clearly understand that the messages, videos, and documents that come before us may or may not be generated by algorithms and not humans. There is a natural preference to know if you are on the receiving end of something that came from a living breathing person or a soulless robot. The state-of-the-art large language models do have specific tells. (For one thing, they don't express themselves creatively as a really smart human can.) It's reasonable to think that an excellent AI-powered sniffer might be able to root out the fakes. But as AI gets better, identifying its output gets harder. Also, once your AI detector figures out the giveaways, those building the models would probably then share those secrets with their products, and an arms race would ensue.

Even if you did have a great way to tell what was algorithm and what was human, it would probably be a bad idea to block the AI stuff. All the companies making productivity apps are providing tools for people to use AI for communications, writing, illustrating, and even video production. You might not like AI, but if you block emails and documents that use it you'll probably miss a lot of meetings and important information.

Instead of labeling which things are made by AI, I think it's more practical to adopt techniques that affirm that something came from actual people. For instance, the Authors Guild (disclosure: I'm on the the council) has recently started a program where books can earn a sticker that says "Created by Humans." Systems like this might help AI-haters like you to limit your consumption to the dwindling percentage of content that's not output from an LLM.

*You can submit questions to mail@wired.com. Write **ASK LEVY** in the subject line.*

End Times Chronicle

When the Northern Lights are seen in the night skies of New Mexico, can we still call them northern?

Last but Not Least

Marissa Mayer explains how she found sunshine after leaving Yahoo.

A key JD Vance adviser touted his addiction to "gas station heroin" and called his boss "a Trump boot licker." Even dumber: He didn't erase his social media posts when he took the job.

National Security Adviser Jack Sullivan is waging a quiet war with China.

Oh no! It's the last episode of WIRED'S Gadget Lab podcast! But don't